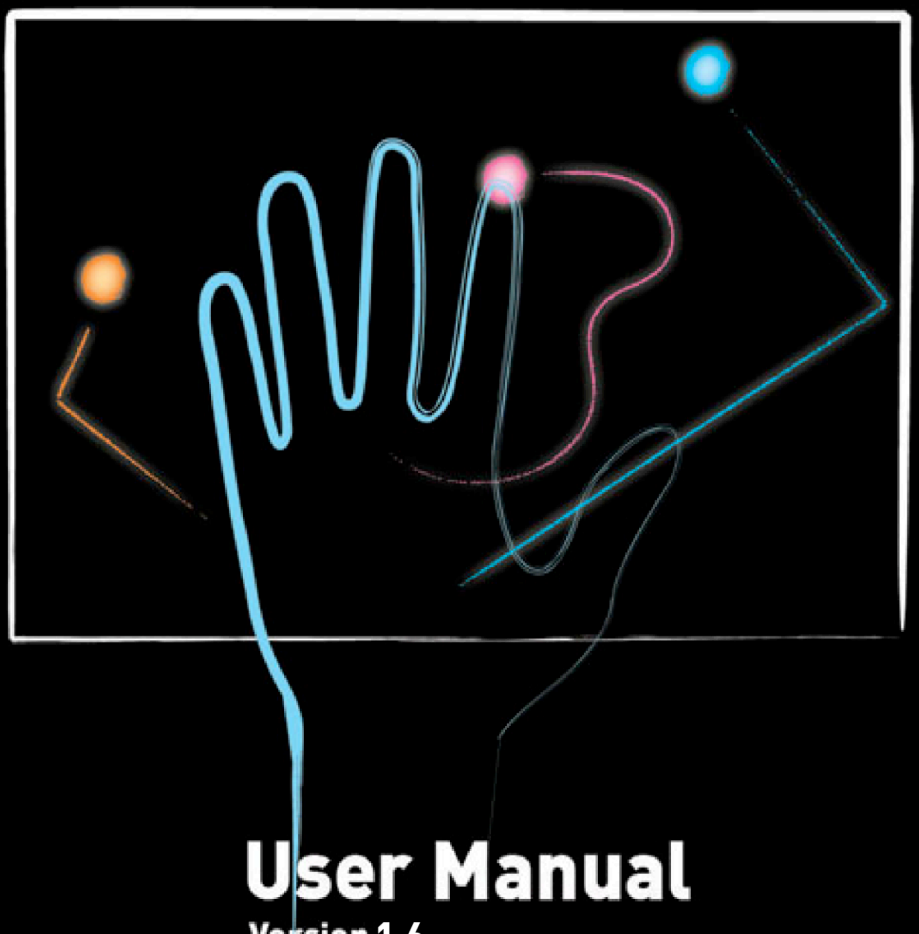
Lemur

The multitouch surface



User Manual

Version 1.6

JAZZMUTANT

www.jazzmutant.com

About the Lemur and this manual

Congratulations for choosing JazzMutant's Lemur! The Lemur is a groundbreaking input device for controlling music and media applications. Its unprecedented modularity combined with its unique multitouch sensing technology will forever change the way you compose, perform, produce and, in more general terms, interact with your work environment.

• What the Lemur is …

The Lemur is a computer peripheral. In other words, it is a device that you connect to a computer.

The Lemur is a controller : its only purpose in life is to provide hands-on control over software running on a computer, or other devices connected to it - such as samplers, synthesizers etc. To interact with them, the Lemur uses two communication protocols : MIDI and OpenSoundControl. Software must comply with at least one of those protocols to be controlled by the Lemur.

• … and what it is not

The Lemur is neither a tablet PC nor a display monitor. That means you cannot write your emails or watch a movie with it.

The Lemur does not produce any sound by itself. There is plenty of wonderful software and hardware out there perfectly doing that job. The Lemur is there to make the use of that software and hardware more efficient and hands-on.

The Lemur is not a Midi Interface; you cannot directly connect it to a master keyboard or any other MIDI device.

In short, the Lemur requires that you own at least one computer running at least one compatible application in order to work. In case you don't, the Lemur is ultimately worthless.

• How to use this manual

This manual aims at leading you through the process of getting a grip on the Lemur and providing you with the technical information you might need to smoothly integrate it into your workflow.

This manual is split into four main parts:

- The first part gives a brief **overview** describing the major features and concepts of your new controller and its software.
- The second parts deals with all things **installation** , and **setup**
- The third part holds the main information needed to work with the Lemur and to create your own interface layouts with the JazzEditor software. You'll find an overview of **all features** including a reference for the available Interface Objects and the Parser.

In the last part you find several **appendices** covering things like the keyboard shortcuts of the JazzEditor, upgrade procedures, troubleshooting tips and documentation of special network configurations

Considering the great variety of applications that can be controlled by the Lemur, it would be nearly impossible to exhaustively detail here all the interfacing possibilities you get from this product. Therefore, instead of providing step-by-step tutorials for every single piece of software you might use, this manual focuses on the general features and concepts one needs to master in order to work with the Lemur. Besides this manual, there exists a host of external material, including additional documentation, templates and didactical workshops meant to make your introduction to Lemur work as smooth as possible. Some of it is available on the provided CD-ROM and a lot more can be download from www.jazzmutant.com.

Last, another great characteristic of the Lemur is the constant evolution of its software, always listening to the Lemur community's needs. For that purpose, JazzMutant regularly releases new updates to both software and documentation. Consequently, the manual you are reading might be already out-of-date. The version covered by this manual is mentioned on its cover. Before getting deeper into the documentation, please make sure that the present documentation matches the software version installed on your Lemur. In general, we suggest keeping both your Lemur and the above documentation up-to-date by regularly checking the product section on www.jazzmutant.com.

Table of Contents

About the Lemur and this manual 1

Table of Contents 4

Chapter 1 Knowing Your Lemur 9

1.1. The Anatomy of your Lemur 9

1.2. Inside your Lemur's brain 10

1.2.1. Projects, Interfaces and Containers 10

1.2.2. Objects 12

1.2.3. Modules 13

1.2.4. Variables 13

1.2.5. Targets 14

1.3. The daily life with your Lemur 14

1.3.1. Cuddling your Lemur 14

1.3.2. Living with your Lemur 15

1.3.2.1. Walking your Lemur 15

1.3.2.2. Your Lemur's Hygiene 15

Chapter 2 Installing your Lemur 16

2.1. Software Installation 16

2.1.1. Mac OS Software Installation 16

2.1.2. Windows™ XP Software Installation 17

2.2. Network configuration – direct connection 19

2.2.1. Three steps 20

2.3. The JazzDaemon 21

2.3.1. The JazzDaemon on Windows™ XP 21

2.3.2. The JazzDaemon on MacOS X 21

Chapter 3 Working With Your Lemur 22

3.1. The JazzEditor Workspace 22

3.1.1. The Toolbar 23

3.1.1.1. Editing and file I/O 23

3.1.1.2. Configuration tools 24

3.1.2. The Project Browser 25

3.1.3. The Editing Area 28

3.1.4. The General Properties 28

3.1.5. The Behavior Properties 29

3.1.6. The Script Panel 30

3.1.7. The OSC Panel 30

3.1.8. The MIDI Panel 30
3.1.9. Workspace Configuration....30
3.2. Interface building 31
3.2.1. Connecting your Lemur....31
3.2.2. The Settings window 32
3.2.2.1. Editor setting....32
3.2.2.2. Lemur Settings....33
3.2.2.3. OSC Settings....33
3.2.3. Creating an interface....33
3.2.4. Creating Objects 34
3.2.5. Saving your project 36
3.2.6. Changing Object appearance 37
3.2.7. Basic Interface Work 37
3.2.7.1. Copy, Paste & Co....37
M 39
3.2.7.2. 39
3.2.8. Configuring Object's behaviour 40
3.2.9. Using Containers 45
3.2.10. Import and Export of Modules 47
3.3. MIDI and OSC Target Setup 47
3.3.1. OSC Settings 47
3.3.2. MIDI Settings....49
3.3.2.1. Lemur MIDI Settings....50
3.3.2.2. MIDI Daemon Settings 51
3.3.2.3. MIDI Setup Example....52
3.3.2.4. Virtual MIDI Ports on Mac....56
3.3.2.5. Virtual MIDI Ports on Windows™ XP 56
3.4. Use your Interface....57
3.4.1. The send mode 57
3.4.2. Setting up OSC mappings....57
3.4.3. Setting up MIDI messages 59
3.4.3.1. Using the MIDI window 59
3.4.3.2. Simple MIDI Mapping....61
3.4.3.3. Multiple Controller Mappings 62
3.4.3.4. Mapping MIDI Note Messages....63
3.4.3.5. Using the Midi Mapping window....64
3.4.4. Trigger Modes....64
3.5. Advanced features 65

3.5.1. The Lemur's memory for Projects 65
3.5.2. Control your Objects with your Objects 66
3.5.3. Making your own Variable 69
3.5.4. Using Vector Variables 72
3.5.5. Using Custom MIDI Messages 74
3.5.6. Bi-directional Control 76
3.5.7. Defining and Using Functions 77
3.5.8. Vector Datacrunching 79
Chapter 4 Object Reference 80
4.1.1. Guide to Object Reference Pages 80
4.1.2. Container 81
4.1.3. Custom Button 81
4.1.4. Fader 83
4.1.5. Knob 85
4.1.6. Leds 86
4.1.7. Monitor 87
4.1.8. MultiBall 89
4.1.9. MultiSlider 92
4.1.10. Pads 93
4.1.11. Range 95
4.1.12. RingArea 96
4.1.13. SignalScope 98
4.1.14. SurfaceLCD 99
4.1.15. Switches 100
4.1.16. Text 101
Chapter 5 Parser Reference 103
5.1.1. Value Colouring 103
5.1.2. Vectors, Singletons and Operators 103
5.1.3. Functions 104
5.1.4. Expressions 104
5.1.5. Accessing Variables 105
5.1.6. Lemur internal clock 105
5.1.7. Lemur Time 106
5.1.7.1. Time Code Example 106
5.1.7.2. LFO Example 107
5.1.8. Operators and Functions Reference 107
5.1.8.1. Arithmetic 107
5.1.8.2. Logic 108

5.1.8.3. Others 109
5.1.8.4. Trigonometric 110
5.1.8.5. Vector 110
Chapter 6 MIDI Message Reference 113
6.1.1. The MIDI Messages 113
6.1.1.1. 80 – Note Off 114
6.1.1.2. 90 – Note On 114
6.1.1.3. A0 – Key Pressure (Polyphonic Aftertouch) 114
6.1.1.4. B0 – Control Change 115
6.1.1.5. C0 – Program Change 115
6.1.1.6. D0 – Channel Pressure 116
6.1.1.7. E0 – Pitch Bend 116
6.1.1.8. F0 – System Exclusive 117
6.1.1.9. F2 – Song Position 117
6.1.1.10. F3 – Song Select 117
6.1.1.11. F5 – Bus Select 117
6.1.1.12. F6 – Tune Request 118
6.1.1.13. F8 – Timing Tick 118
6.1.1.14. FA – Start Song 118
6.1.1.15. FB – Continue Song 118
6.1.1.16. FC – Stop Song 119
6.1.1.17. FE – Active Sensing 119
6.1.1.18. FF – System Reset 119
Chapter 7 Appendices 120
7.1. Network Configuration Guide 120
7.1.1. Glossary 120
7.1.2. Computer directly connected to a Lemur (as known as P2P) 120
7.1.2.1. Computer configured as a DHCP Server – Connection Sharing 120
7.1.2.2. Computer and Lemur using DHCP 120
7.1.2.3. Computer and Lemur using Static IP 120
7.1.3. Computers and Lemur connected to a Network 121
7.1.3.1. Network with a DHCP Server 121
7.1.3.2. Network without DHCP Server 121
7.1.4. Lemur Network Setup 121
7.1.5. MacOS Network Setup 122
7.1.6. Windows™ XP Network Setup 126
7.1.7. Lemur – Computer network communication 129
7.2. Shortcuts 130

7.3. Upgrade your Lemur....131
7.3.1. Lemur hardware settings....132
7.4. The FTP Server....132
7.5. Warranty....134
7.5.1. Warranty and Liability....134
7.5.2. Your Rights....135
7.5.3. Applicable laws and court of jurisdiction....135

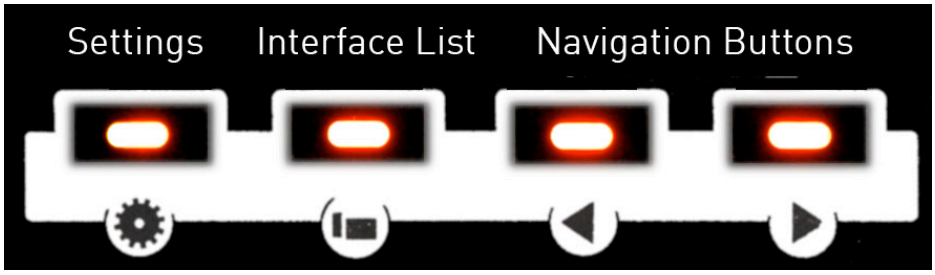
Chapter 1 KNOWING YOUR LEMUR

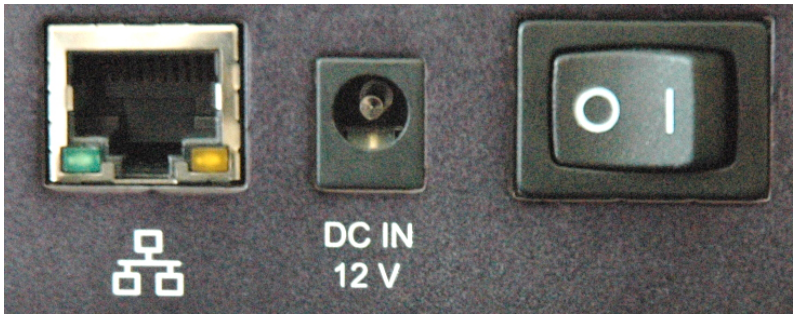
Not only does the Lemur look unlike any controller out there, it also introduces new ways of describing human-machine interaction. This chapter will put you in acquaintance with your new pet and the way it thinks.

1.1. The Anatomy of your Lemur

Let's first have a look at your Lemur's anatomy. The top of the Lemur obviously features the Multitouch screen, consisting of the touch-sensitive surface and the ultra-bright TFT Display. The first allows you to manipulate colourful user interface objects displayed on the latter with all your ten fingers independently. Above the display, there are four buttons which have different utility functions :

- The **Settings** button gives your access to the Lemur network, OSC (see chapter 3.3), MIDI (also to be found in chapter 3.3), display and touch panel configuration. It also displays the version number of the firmware running on your Lemur.
- The **Interface List** button displays a gallery of thumbnails depicting each interface currently stored on the unit. Touch the desired interface to display it full screen.
- The **Interface List** button also contains the link to the **Lemur Project browser**.
- The two **Navigation** buttons allow you to move forwards and backwards through the Interfaces currently stored on the unit.





On the **rear panel** of the Lemur you find the power switch and the power input jack.

Only connect the original power supply that came with your unit as it is perfectly verified for providing the correct currents.

Next to the adapter input, there is an Ethernet jack with two status lights. This is the Lemur's door to communication with the outside world. Out of all the existing manners to connect the Lemur to your computer, the easiest one consist in plugging the provided network cable between the Lemur Ethernet Input and your computer Ethernet interface.

NOTE : The **Ethernet** jack can be used with any network cable. Please use CAT5 or better quality. Lower grades will not work properly. Depending on the situation you will either need a **crossover** network cable (for direct connection to the computer - that's the cable that came with your Lemur) or a **non-crossover** network cable (for connecting the Lemur to a hub or switch; this situation is described in the Appendix **Network Configuration Guide**).

1.2. Inside your Lemur's brain

The Lemur introduces a new paradigm in the realm of media control, and along with this brand new approach, its usage might slightly differ from the way you are used to deal with conventional controllers. Therefore, before you start working with your new controller, we felt it was worth making you familiar with the core concepts of the system.

1.2.1. Projects, Interfaces and Containers

As you might have noticed, the main characteristic of the Lemur is that it doesn't feature any physical control object, such as real pads, keyboard, bulky faders or joysticks. Instead, there is a blank touch panel that your can populate with as many different virtual objects as you need. This way, you can design the interface layouts that will perfectly fit both your application and your hands. For

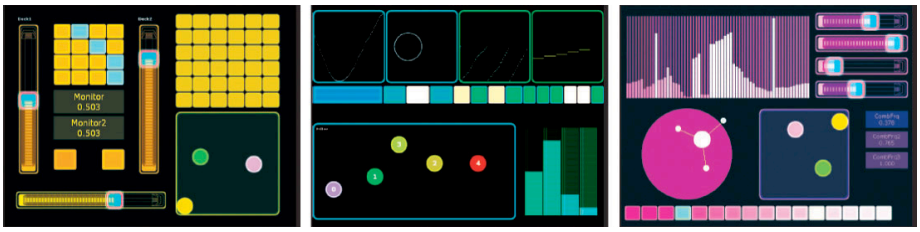
that purpose, the Lemur is provided with a dual-platform (MacOSX and Windows XP) interface design utility, the mighty **JazzEditor.**

This application lets you import or create complete control templates for the Lemur, which we call **Projects**. All Project files the JazzEditor produces are stored somewhere on your computer's hard drive as XML files.

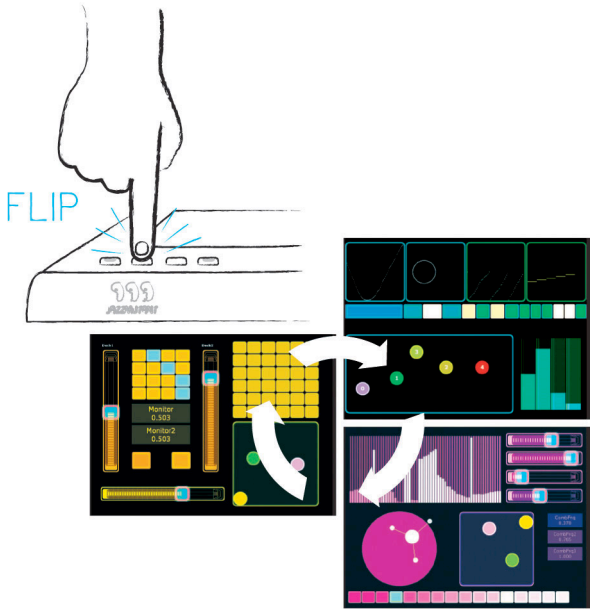
Furthermore, the Projects can be stored in the Lemur's internal **Flash memory**. You can access the **Lemur Project browser** with the second **button** on your Lemur and a touch of the Project Browser button. You are presented with a list of stored Projects and with commands for opening, saving or deleting Projects residing on the unit.

The Lemur features a small **FTP server** for shuffling projects back and forth between the Lemur and your computer. Please refer to chapter 7.4 for details.

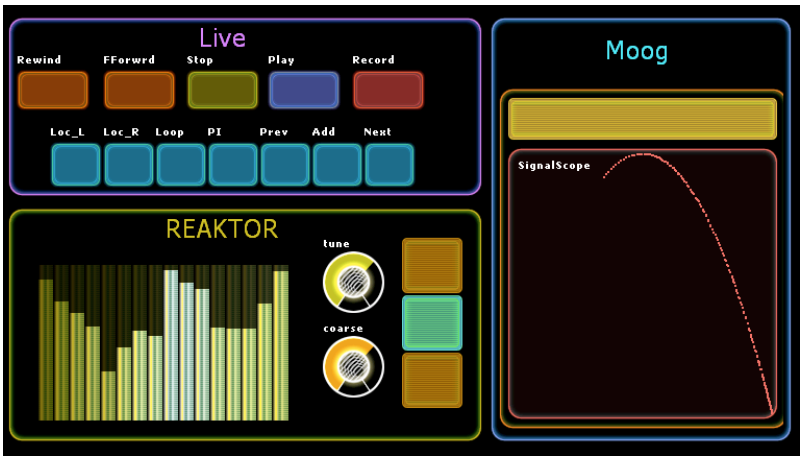
A Project can contain many **Interfaces**. You can think of them as pages in your Project containing multiple **Objects** that can be organized in **Containers**. The number of Interfaces and Objects is limited only by the internal memory of the Lemur, which gives you plenty of room for your creations. The figure below shows a typical Lemur project, comprised of three different interfaces.



Once a project is loaded on the Lemur, the different Interfaces can be browsed through via the **Interface List** button or switched to with the **Navigation** buttons. The Interface List button presents them as an array of small icons that you can activate by touch. Use the Navigation buttons to step through the Interfaces one by one, as shown below.



The Containers let you organize the Objects on an Interface. Containers have a **frame** around them leading to tidy looks and easy structuring of your control surface. They can contain multiple Containers themselves building a hierarchy of Interface elements. The picture below shows an interface featuring three different containers.



You can lock a Container and its content will become un-editable and will also vanish from the Project Browser. This improves the overview when working on large Projects.

All details regarding projects, interfaces and containers are disclosed in chapter 3 : Working With Your Lemur

1.2.2. Objects

Objects are the main message generators of the Lemur : they provide the values you control with your fingers. There is a multitude of different Objects available that all have their special capabilities:

- Custom Button
- Fader
- Leds
- Knob
- Monitor
- MultiBall
- MultiSlider
- Pads
- Range
- RingArea
- SurfaceLCD
- SignalScope
- Switches
- Text

Each particular object comes along with a set of specific options enabling you to customise their appearance (e.g., size, color, and brightness) and their behaviour, i.e. the way they respond to your fingers. Objects can send and/or receive a plurality of Midi or OSC messages from your applications. Different Objects on the same Lemur Interface can be connected to different applications and even to different computers on the network (see below for details on OSC and MIDI Targets). One object can even be used to modify in real-time the properties of another object. JazzEditor provides a set of handy tools to easily map objects output to the software you wish to control and to create ultimate interactions between different objects.

For more information about the different Interface Objects and how to work with them, please have a look at chapter "Interface Building" and at the Object Reference further down the text.

1.2.3. Modules

Additionally, you can import pre-built parts of project as **Modules.** A Module is a set of reusable interface elements. A module can feature objects, containers and even interfaces. A library of modules is provided on the Installation CD, and more modules are available from JazzMutant's website.

You can also create your own user library of reusable items. Indeed, any part of a Lemur Interface can be exported as a **Module** for future use in other Projects. Due to this feature you never have to develop the same thing twice. Just save your building blocks as Modules to produce a growing library of re-useable, object-oriented Interface parts. The User Library on the JazzMutant website also is a great resource for stacking up on handy Interface elements.

1.2.4. Variables

Variables are the mathematical basis of the Object's output. Any Object has its own set of Variables that change when you touch it with your fingers. Most object properties are also variables. They can be changed by external software or by the status of other Lemur Objects dwelling in the Project. Those variables are called **built-in Variables** because you always find them with the respective Object.

You can also create your own **user-defined Variables** that you can fill with your own Expressions.

Variables can be **local**, living in a specific Object and accessed from the outside through its address. You can also define them **globally**, enabling multiple Objects to use their values directly.

Variables can be **multi-dimensional**, too. A Fader, for example, has only one Variable defining its output with a single number. A MultiBall Object has three Variables for the X, Y position and the lightness of the balls. The MultiBall Variables are Vectors with as many components as there are balls defined for the Object.

In combination with the built-in **Functions** and **Operators** and the user-definable **Expressions** the Variables become a powerful tool for refining Object behaviour, scaling and shaping responses, and setting up interactions

between Objects. Please refer to the section about Interface building and the respective reference chapters for more information.

The global **Time Variable** deserves a special mention. It presents the time in milliseconds since the Lemur was switched on. Combined with the Lemur's mathematical operators, it can produce a great variety of time-varying number sequences.

1.2.5. Targets

The networkability of the Lemur endows it with the unique capacity to simultaneously control several applications running on the same computer, and even to control several pieces of software launched on different computers. Each application or device controlled by the Lemur is called a **Target**.

Each of Object's Variables can be mapped to any of the eight available **MIDI Targets** and to any of the eight **OSC Targets**.

A MIDI Target is a pair of MIDI In- and Outputs and an OSC Target is characterized by its IP address and its port number. This means a Target can be software on a networked computer or a piece of MIDI kit you have sitting in your rack. No matter how complex your setup may be, you can control everything from a single Interface on your Lemur by individually assigning the different Variables to different Targets.

The **JazzDaemon** (see chapter 2) makes MIDI routing as flexible as Network routing. You can connect to the **MIDI ports** of any computer on the network that has the JazzDaemon running. Use the **MIDI port browser** (see Chapter 3.2.2) on your Lemur to get a list of running JazzDaemons on the network and to connect to specific MIDI ports on the respective machines. This makes it easy to control with a single Lemur various applications and MIDI hardware even if they are running on different computers.

Because the MIDI and OSC Targets can be chosen on a **per Object basis** you can integrate all your applications and MIDI hardware into a single Lemur interface.

1.3. The daily life with your Lemur

1.3.1. Cuddling your Lemur

The Lemur loves to be touched. After all, it is its main purpose in life. But as it's the case with all primates, the nature of the touch will affect the outcome.

The multitouch technology used in the Lemur combines high resolution with high sensitivity. You don't have to use excessive force to control the objects on your interfaces. Extremely light touch, on the other hand, won't work either. Just practice a bit with your Lemur and you will quickly find the right balance combining precision with tactile feedback.

The best technique is using the fingertips and not the flat underside. It leads to a maximum of precision and ensures good agility for the various multi-finger gestures you will develop when working with the Lemur.

1.3.2. Living with your Lemur

There are a few elementary rules that need respecting carefully if one want to have a pleasant and durable life with one's Lemur.

1.3.2.1. Walking your Lemur

When transporting the Lemur, please ensure it is properly packed to protect it from shock and damage. A good idea is using a sturdy laptop case (15" or 17" should fit). Place an additional protective layer over the display.

There are two types of power cables enclosed with your Lemur. One is of the European type and one works in the USA. If you take your Lemur abroad to other countries, please make sure you get the appropriate power cable for the respective country.

1.3.2.2. Your Lemur's Hygiene

The Lemur is a sturdy creature. If you follow some basic guidelines concerning care and feeding it will be your companion for years and years to come.

For everyday cleaning wipe the unit with a soft, dry cloth or one that has been slightly dampened with water. To remove stubborn dirt, use a cloth with a mild, non-abrasive detergent. Afterwards, be sure to wipe the unit thoroughly with a soft, dry cloth.

Never use gasoline, thinners, alcohol or solvents of any kind. Chemicals may cause fading of colors and/or deformation of the surface.

When working with the unit's buttons, display, or other controls or when using its cords and cables try to be reasonably gentle. Rough handling can lead to malfunctions.

Never strike or apply strong pressure to the display.

When connecting/disconnecting all cables, grasp the connector itself. Never pull on the cable. This way you will avoid causing shorts, or damaging the cable's internals.

Use the Ethernet cable supplied with the unit. If using some other cable, please make sure it is of the correct type. When you connect the Lemur directly to a computer it has to be a crossover Ethernet cable. The JazzMutant Ethernet cable you received with your Lemur is of that type. If you connect your Lemur to a hub or switch, please use a high quality, non-crossover Ethernet cable (CAT5 or better).

Chapter 2 INSTALLING YOUR LEMUR

2.1. Software Installation

For the Lemur to become functional you have to install the JazzEditor software that enables you to create custom interfaces and that also manages the connection to the Lemur. You find the **JazzEditor** Installer on the provided Installation CD.

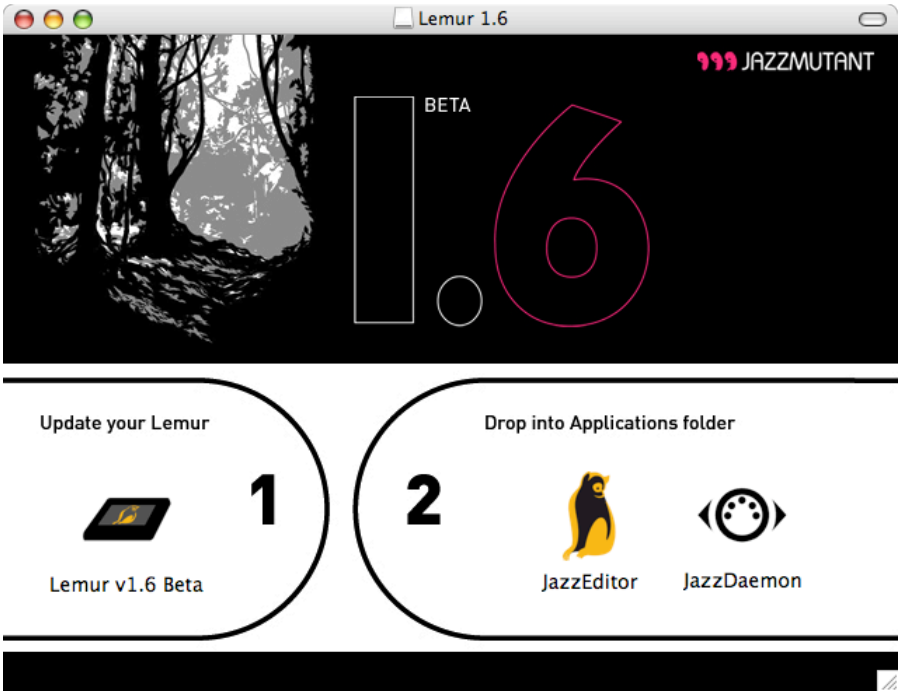
Also included is the **JazzDaemon**, a small application running in the background that handles the MIDI data flow from and to the Lemur via the network. It is necessary to achieve any kind of MIDI communication using the Lemur.

2.1.1. Mac OS Software Installation

Software & Hardware Requirements Mac:

- Power G3 or earlier based computer
- CD/DVD drive to install from the provided CD
- MacOS 10.4 (Tiger) minimum
- Network Adapter (Ethernet/Wifi/Airport/USB/Firewire)

Double-click on the **Lemur v1.6.dmg** file located on the provided CD.



Open the **Lemur v1.6** updater if your Lemur runs and older version.

Drop the **JazzEditor** and **JazzDaemon** applications in your **Applications** folder.

2.1.2. Windows™ XP Software Installation

Software & Hardware Requirements Windows™:

- P3 800 MHz or greater, 64 Mo SD-RAM
- Windows XP SP 2
- CD-Rom Drive
- Ethernet Interface

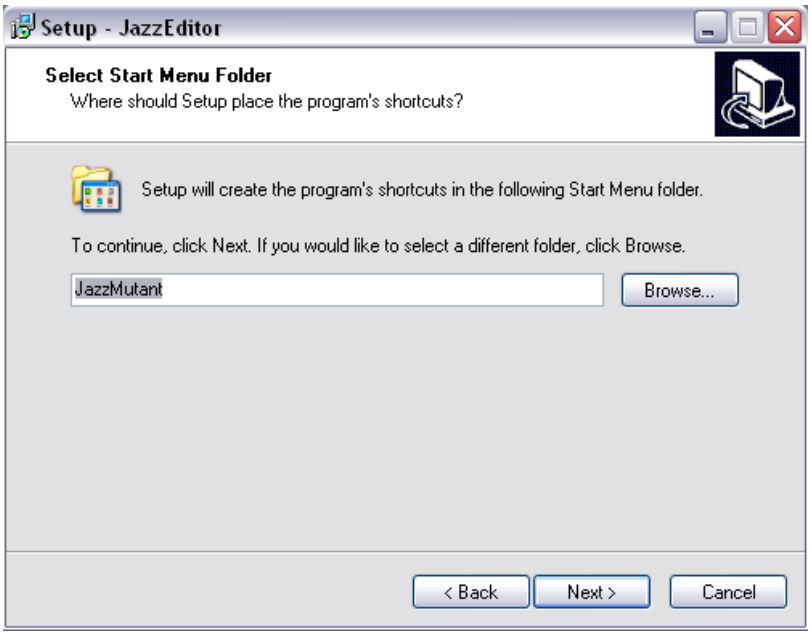
For installation of the software open the CD in the Windows Explorer and double-click the JazzEditor_setup.exe.

Setup - JazzEditor
Welcome to the JazzEditor Setup Wizard
This will install JazzEditor version 1.5 Beta on your computer.
It is recommended that you close all other applications before continuing.
Click Next to continue, or Cancel to exit Setup.
Next >
Cancel

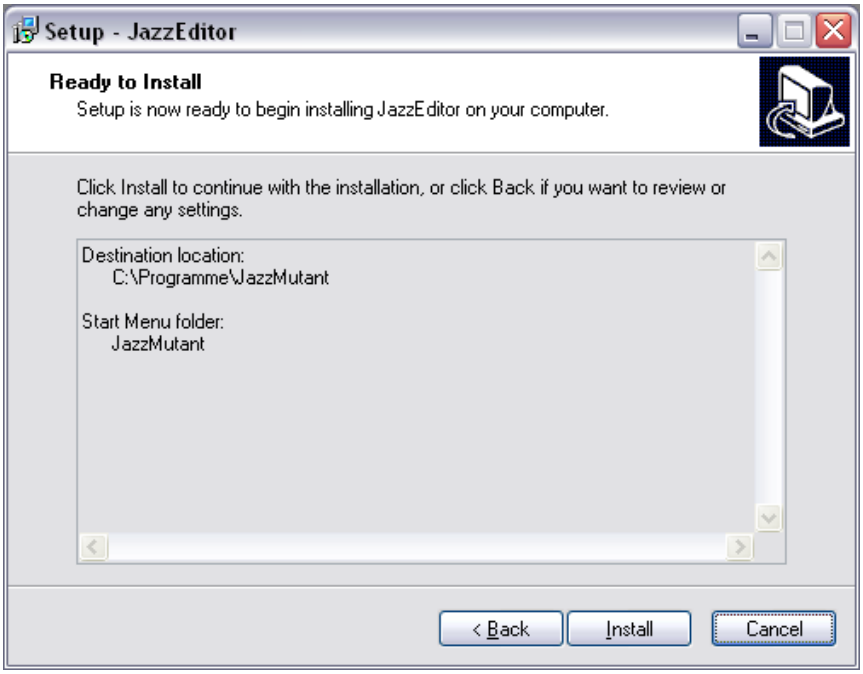
Click on Next to start the Installation.

Setup - JazzEditor
Select Destination Location
Where should JazzEditor be installed?
Setup will install JazzEditor into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Programme\JazzMutant
Browse...
At least 1,1 MB of free disk space is required.
< Back
Next >
Cancel

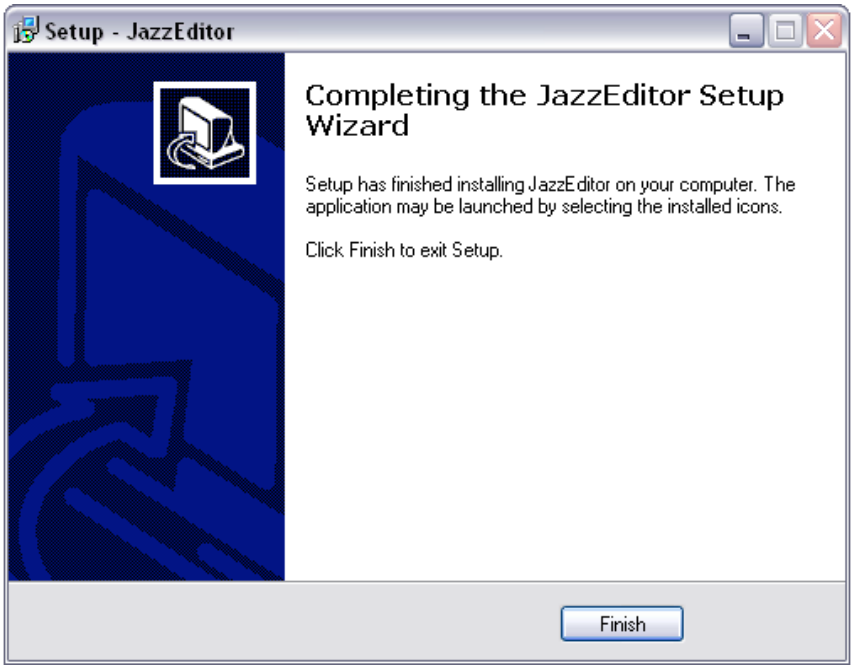
We recommend using the default installation path (on English systems it is C:\Program Files\JazzMutant). If you have your own way of organizing things you can also choose a different folder.


Setup - JazzEditor
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
JazzMutant
Browse...
< Back
Next >
Cancel

In this step you can choose the name of the Start Menu folder where Windows will put the shortcut for starting the JazzEditor.


Setup - JazzEditor
Ready to Install
Setup is now ready to begin installing JazzEditor on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
C:\Programme\JazzMutant
Start Menu folder:
JazzMutant
< Back
Install
Cancel

The next screen summarizes the settings you chose before. If you click on Install the software is transferred to your hard disk and all necessary are made automatically.


Setup - JazzEditor
Completing the JazzEditor Setup Wizard
Setup has finished installing JazzEditor on your computer. The application may be launched by selecting the installed icons.
Click Finish to exit Setup.
Finish

Just click Finish on the last screen of the installation procedure and you are done.

2.2. Network configuration – direct connection

The Lemur uses standard Ethernet to communicate with your computer or with other devices on the network. You can connect it directly to your computer

with a single, crossover Ethernet cable, or place it on a local area network via a hub or switch and a non-crossover Ethernet cable.

You may want to use a local area network if any of the following are true:

- You are using your computer's Ethernet port to connect to the internet, and you want to keep the connection while working with your Lemur.
- You use a WLAN connection to the internet via a wireless router, and you would like to access your Lemur wirelessly by connecting it to your router.
- You have several Lemurs you would like to use in parallel (Lemurs are very social creatures – one is never enough).

You may want to connect your Lemur directly to you computer with a single cable if any of the following are true:

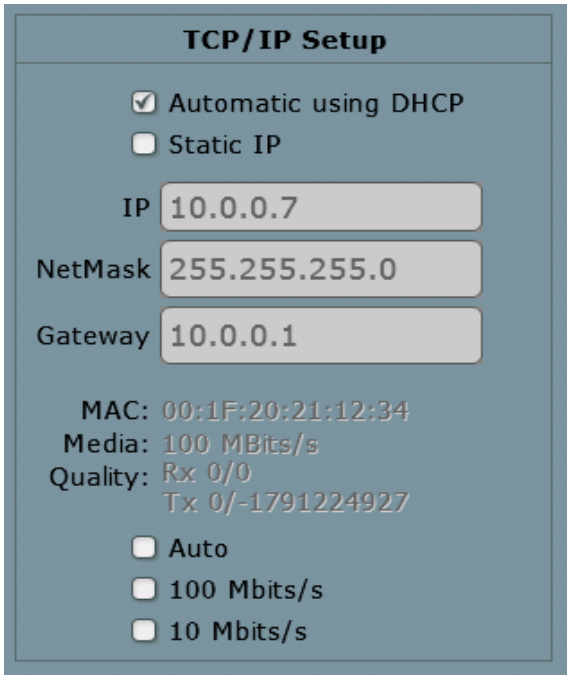
- Your Lemur is the only Ethernet device you have (either because you don't own any others or you connect wirelessly to other Ethernet network devices).
- You are looking for the simplest possible configuration for live performance.

To get in business rapidly, we'll only cover the latter configuration in this section. It should suffice on most systems. Should it not work out of the box, or if you seek more details on OS tinkering and connection to hubs and switches, please have a look at the Appendix on Network Configuration.

2.2.1. Three steps

The direct connection procedure basically boils down to three steps.

- Connect the Lemur's Ethernet port to your computer's one, using the provided crossover cable.
- Make sure any firewall software is deactivated on your computer.
- Turn your Lemur on and press the settings button. On the appearing panel, just select the **Automatic Using DHCP** mode.



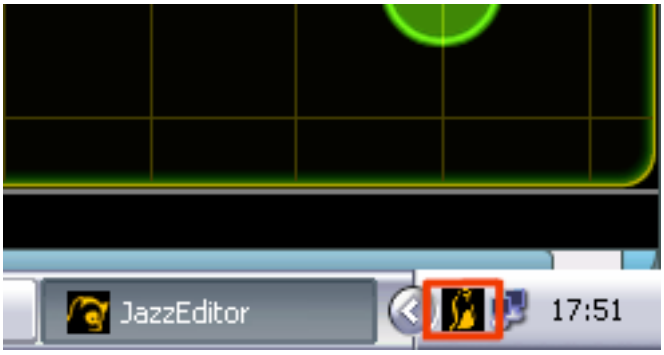
2.3. The JazzDaemon

The **JazzDaemon** is a little helper application running in the background on your system. It handles the MIDI data flow between the Lemur and the MIDI ports installed on the computer.

The Daemon automatically scans all available **MIDI ports** on the computer. These may be physical hardware ports or virtual ports like offered by MIDI Yoke or Maple. The detected ports are then made available to the Lemurs on the network.

2.3.1. The JazzDaemon on Windows™ XP

By default the Daemon is loaded automatically at boot time of the computer. You can find it in the System Tray of your Windows™ XP desktop.



A click on the Daemon opens a menu with two entries:

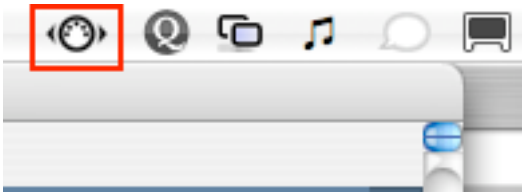
- **Setup MIDI Daemon** opens the MIDI port browser of the Daemon. Please have a look at chapter 3.3 for details.
- **Quit** shuts the MIDI Daemon down.

Please note that the Daemon **must be running** if you want to use MIDI with the Lemur. Use the small **X** in the upper left corner of the MIDI connection browser for closing it without shutting the Daemon down.

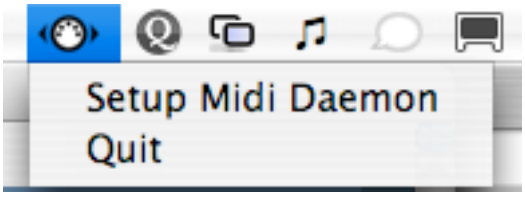
If you don't want the Daemon to be automatically started, you can change its behaviour. Uncheck the **Launch Daemon at Startup** flag and the next time you boot your computer the Daemon won't be started.

2.3.2. The JazzDaemon on MacOS X

The Daemon doesn't appear in your Dock while it's running. You can find it on the Menu Bar of your MacOS X desktop.



A click on the Daemon opens a menu with two entries:



- **Setup MIDI Daemon** opens the MIDI connection browser of the Daemon. Please have a look at chapter 3.3 for details.
- **Quit** shuts the MIDI Daemon down.

Please note that the Daemon **must be running** if you want to use MIDI with the Lemur. Use the small **X** in the upper left corner of the MIDI port browser for closing it without shutting the Daemon down

If you want the Daemon to be automatically started, you can enable this by enabling a single checkbox. Choose Setup MIDI Daemon and click the **Preferences Tab**.

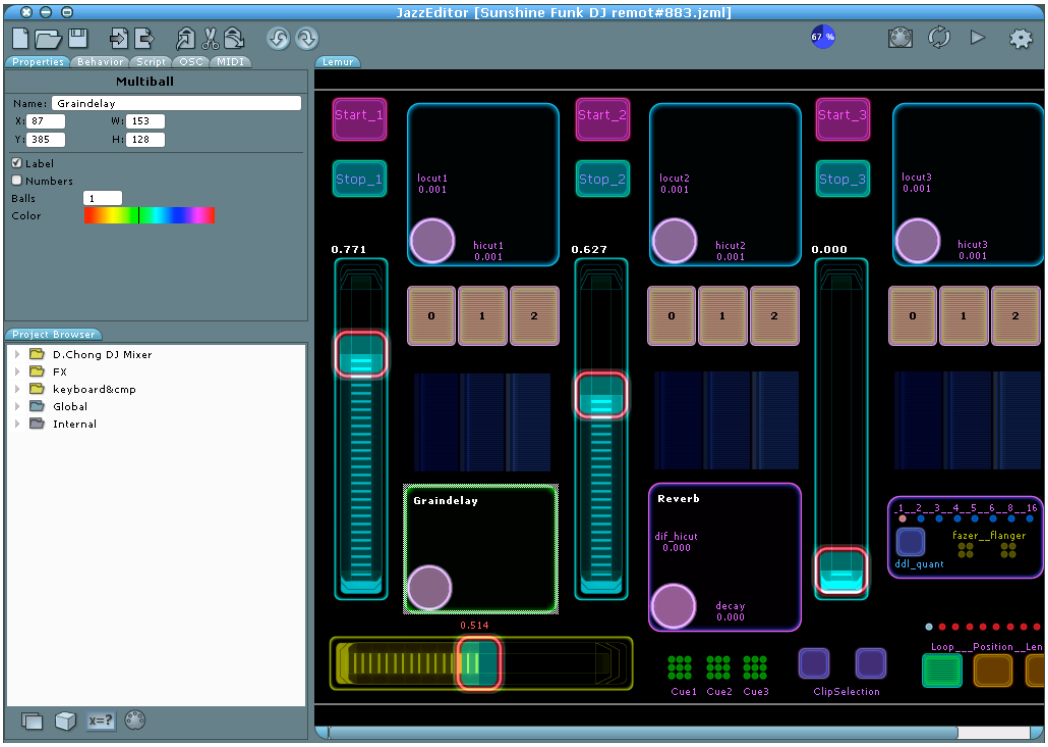
Chapter 3 Working With Your Lemur

We are now ready to start working with the Lemur. This chapter will guide you through the use of the JazzEditor application and introduce you to the workflow involved with designing Lemur control interfaces.

3.1. The JazzEditor Workspace

All editing and building of Interfaces is done via the JazzEditor Software. Here you create, modify and save your Projects, as well as manage connections to your Lemur and to external gear via Ethernet and MIDI.

This section will walk you through the different parts of the Workspace. Please read the section about Interface building for details on how to actually use them and refer to the Object Reference for complete documentation of all parameters.



The JazzEditor **Workspace** is your home when it comes to building Lemur Interfaces. It consists of four major areas.

- The **Toolbar** on top of the Workspace provides typical items you might find in an application's File and Edit menus.
- The **Editing Area** is the place where the actual Interface Objects are placed to design Lemur Interfaces.
- The **Selection specific Panels** allows for editing details and mappings of selected interfaces, objects, variables, etc.
- The **Project Browser** provides a hierarchical view of your current project (Interfaces, Objects and Variables).

A lot of the commands and functions of the Workspace can also be controlled via keyboard shortcuts. The shortcuts are displayed in the Tool tips that pop up when you hover with your mouse pointer above the respective command. Please have a look at the corresponding appendix for a complete list.

3.1.1. The Toolbar

3.1.1.1. Editing and file I/O

On top of the JazzEditor you find the elements of the **Toolbar**. To the left you see buttons dealing with Project creation, in- and output to disk and the undo system.

- The **New Project** Button creates an empty Project.
- The **Open Project** command presents you with a dialog for opening Projects from your file system.

- The **Save Project** button opens a dialog for saving your current Project to the file system. Choose an appropriate folder to store the Project. Please note, that the **Save As** command is reachable via using the corresponding keyboard shortcut (Control+Shift+S for PC and Command+Shift+S for Mac).
- The **Import Library** and **Export Selection** commands are for import and export of **Modules**. Any imported Module is incorporated into the currently opened Interface at the level of hierarchy you choose and you can select any group of Objects for export as a Module to the file system. Please note that you can also import and export Interface elements via contextual menus that can be triggered in many parts on the user interface.
- Use the **Copy**, **Cut** and **Paste** commands as you know them from other applications. They are also available via contextual menus.
- The last two commands on the left side of the Toolbar are the **Undo**/**Redo** buttons. You can roll back an unlimited number of JazzEditor commands with this system.

3.1.1.2. Configuration tools

The Toolbar also contains commands for configuring the Lemur.

To the left you have the **Memory Display** showing you the percentage of the Lemur's memory that is occupied by your Project. This is for reference, only.

The **MIDI Mapping** button opens a window with all MIDI mappings used in the current Project.

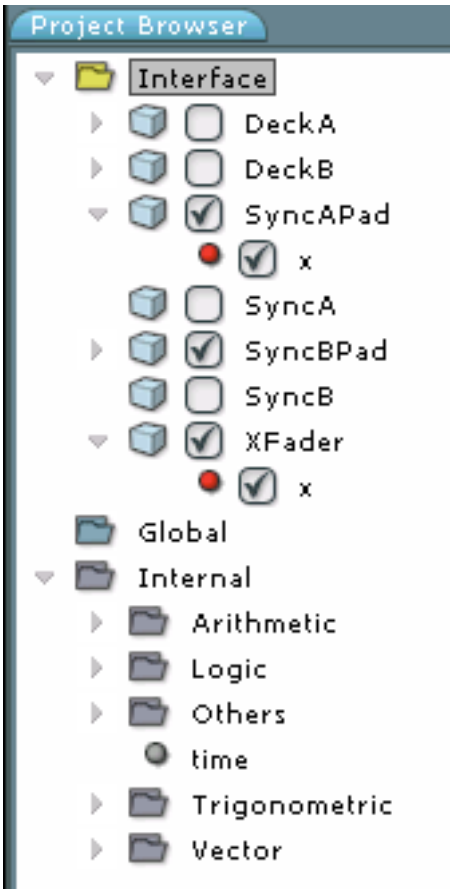
If you activate the Lemur **Synchronization** button, any changes to Objects' states on the Lemur will also be reflected in the JazzEditor. This can be very handy for saving a Project including the state of all Objects. **One-shot synchronization** can be achieved via the F5 key on your computer keyboard.

The Lemur Synchronization of course only works if you have the same Project running on your computer and on the connected Lemur.

The **Connection** button opens a window listing all the accessible Lemurs on the network. This is the place where you create a connection between the JazzEditor and a Lemur.

The **Settings** button opens a window that let you configure OSC targets (if there is a Lemur connected), as well as upgrade a connected Lemur.

3.1.2. The Project Browser

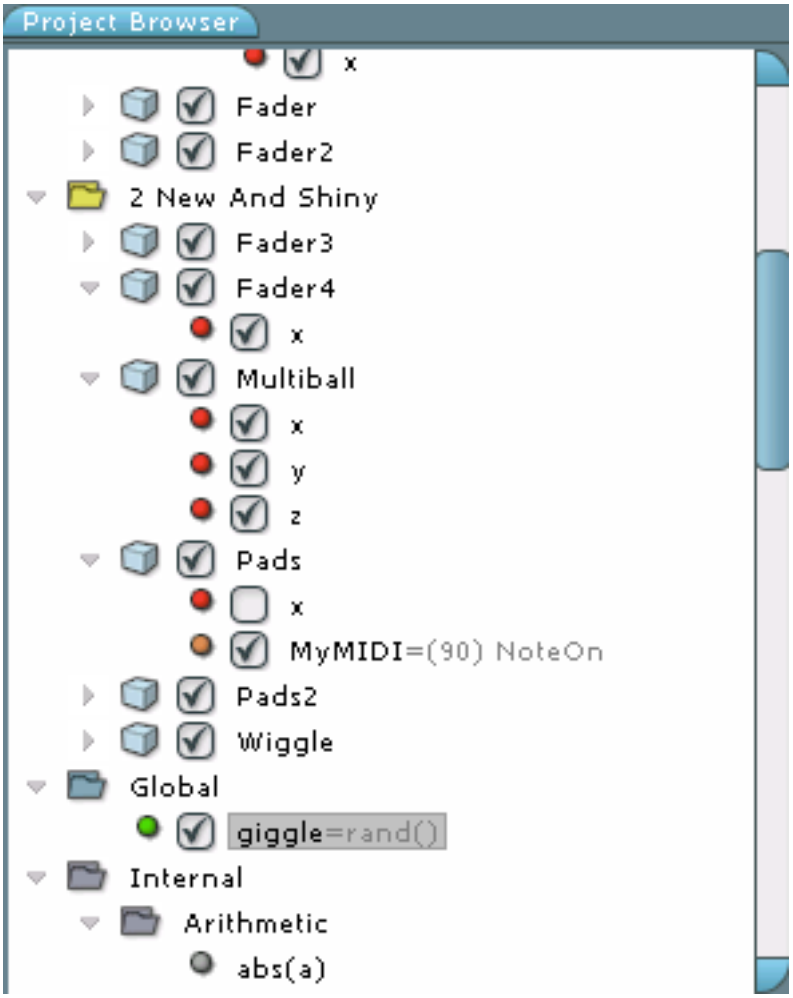


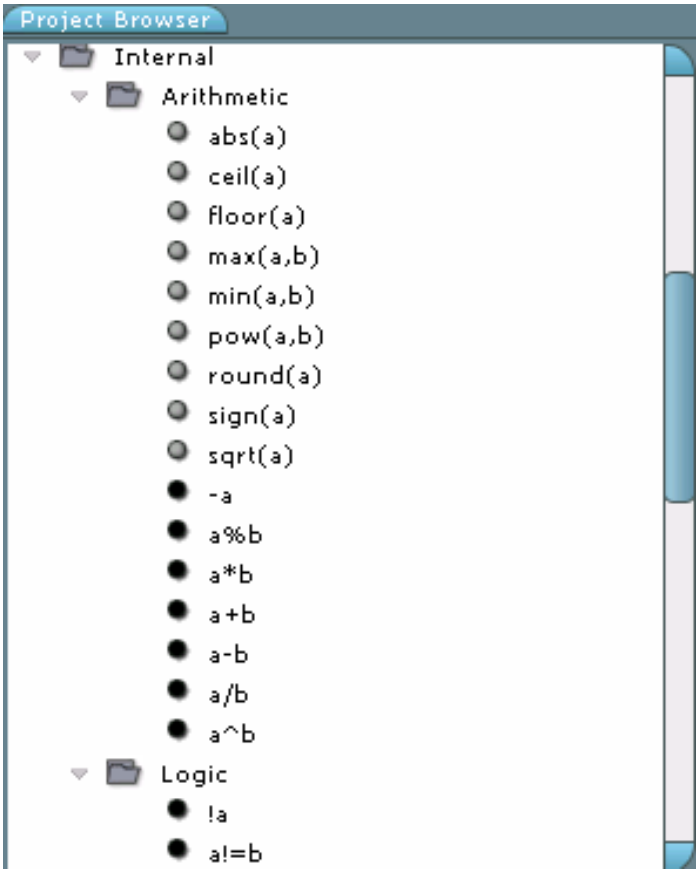
In the lower left part of the Workspace you find the **Project Browser**. It displays all elements of the current Project in a hierarchical tree structure. If you need more space to see all levels of your Project just grab the top or right border of the Project Browser and drag it to resize the browser.

Click on the small **disclosure triangles** in front of the symbols or double-click on a symbol to **expand** or **collapse** the next deeper level of the tree.

The upper part of the tree displays the **Interfaces** of the Project.

Inside the Interfaces are **Objects** and their **Variables**. The small **checkboxes** play an important role in Lemur Projects. If a box is checked it means that the respective Variable is transmitted via OSC or MIDI. If it's unchecked it is only used internally (e.g. as a reference for other Variables). Global send boxes on Objects and Containers let the user activate/deactivate OSC and MIDI mappings for a whole hierarchical level of a project.


Project Browser
x
Fader
Fader2
2 New And Shiny
Fader3
Fader4
x
Multiball
x
y
z
Pads
x
MyMIDI=(90) NoteOn
Pads2
Wiggle
Global
giggle=rand()
Internal
Arithmetic
abs(a)



Below the Interfaces you find the **Global Variables** and the internal **Operators** and **Functions**. These are just for your reference and you cannot interact with them using your mouse. Find out more about Operators and Functions in the Parser Reference section.

The various entries in the Project Browser are color-coded.

- **Red** dots stand for the Object's main parameters (x for a Fader; x, y, z for a MultiBall, etc.)
- **Orange** dots depict a Custom MIDI message
- **Green** dots show Expressions created by the user.
- **Blue** dots show Functions defined by the user.
- **Grey** stands for internal Functions
- **Black** dots are the Operators

Use the four **Create** commands to create new **Interfaces**, **Objects**, **Expressions** and **Custom MIDI mappings**.

3.1.3. The Editing Area

The Editing Area is where you actually construct the Interfaces. It always displays the Interface that is currently selected in the Project Browser.

Any freshly created Objects appear in the upper left corner of the current Interface. Fetch them with your mouse and move them to the spot you want.

If you hover with your mouse over an Object its corners highlight. Grab a corner and move the mouse to resize the Object.

Objects can overlap on the Interface. The rule of thumb is that the Object created last will cover older Objects. Some Objects (like the Monitor) can be transparent leading to various possibilities for labels and captions. When the Transparency flag is activated, only parts of the Object will remain opaque and any Objects lying underneath can be seen.

Multiple Objects can be selected by holding the Shift key and clicking on them one after the other. You can also drag a square around them to get the same effect. By clicking an Object again while still holding the Shift key you remove it from the selection.

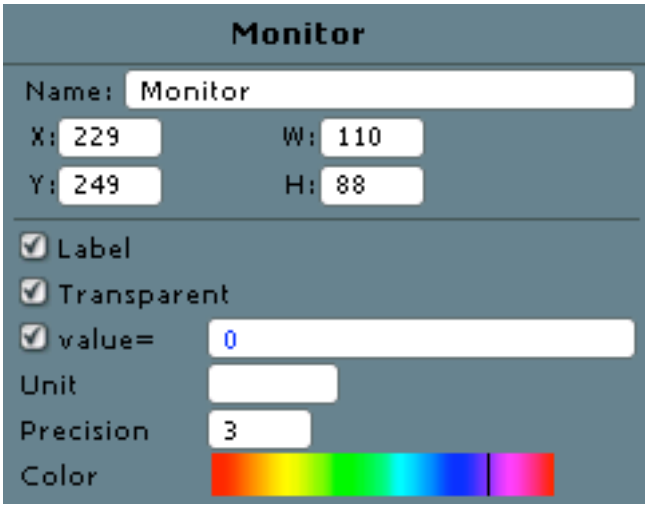
There are two different modes available concerning interaction with Objects in the JazzEditor. In default Edit mode you cannot interact with Objects. In Run mode you can use them with your mouse as you would do it with your fingers on the Lemur. Simply press "e" on your computer keyboard and hold it to switch to Run mode. As long as you hold the key you can use your Interface. This comes very handy when debugging your Setup without having your Lemur connected.

The Editing Area also features a contextual menu. Open it with a right-click (command-click for Macs). It offers an additional way for executing the following commands for working with Objects or groups of them:

- Copy, Cut, Paste, Delete
- Group/Ungroup
- Import/Export

3.1.4. The General Properties

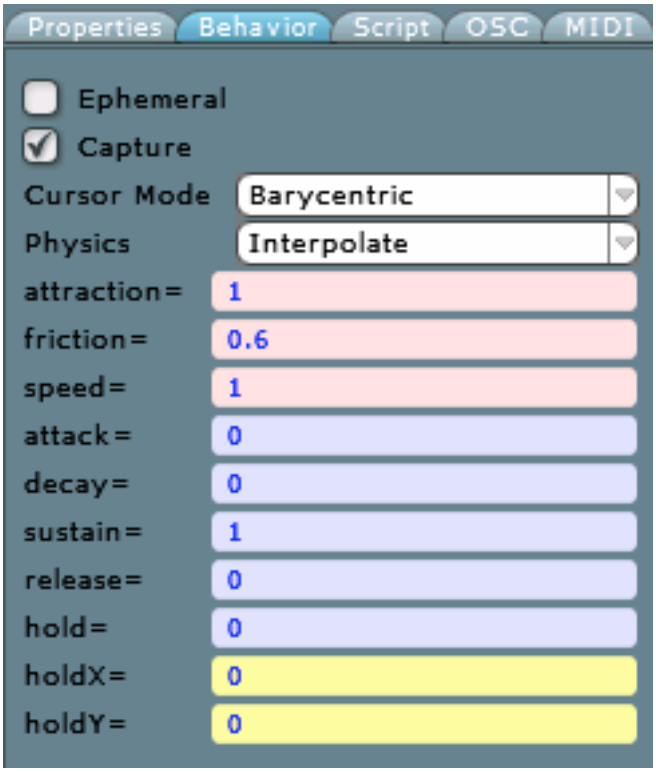
The Properties panel provides access to the basic properties of the Lemur's Objects. As the parameters vary from Object to Object, the details are given in the Object Reference chapter.



The General Properties deal with appearance and Variables. Here you find number-fields for positioning and resizing of Objects. Depending on the Object the additional parameters may vary. Please look at the Object reference for information on Object-specific parameters.

The name of Objects on the Lemur can be up to 64 characters long.

3.1.5. The Behavior Properties

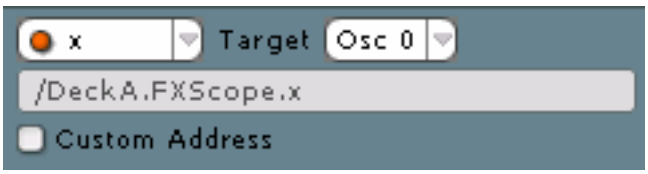


The Behavior panel circles around the physical characteristics of the Objects. Here you find parameters like friction and tension and the different physics modes of the Object. We'll cover those in the chapter Using your Lemur and in the Object reference.

3.1.6. The Script Panel

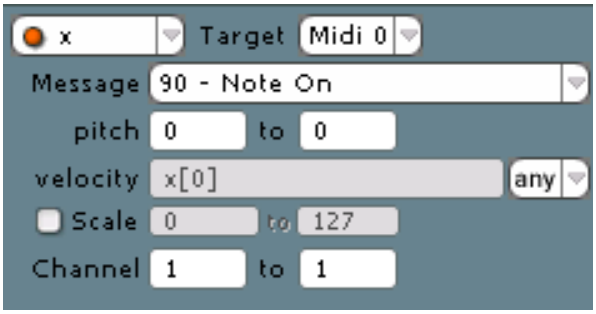
The Script panel is for defining expressions for Variables and Functions, be it globally or locally. Have a look at Chapters 3.3.2 and 3.5 for details on forging your own expressions.

3.1.7. The OSC Panel



On the OSC Panel you can define the routing of the different Variables to OSC Targets. You have a menu for the Variables and one for the eight possible OSC Targets. This panel also serves for controlling Lemur objects via OSC.

3.1.8. The MIDI Panel

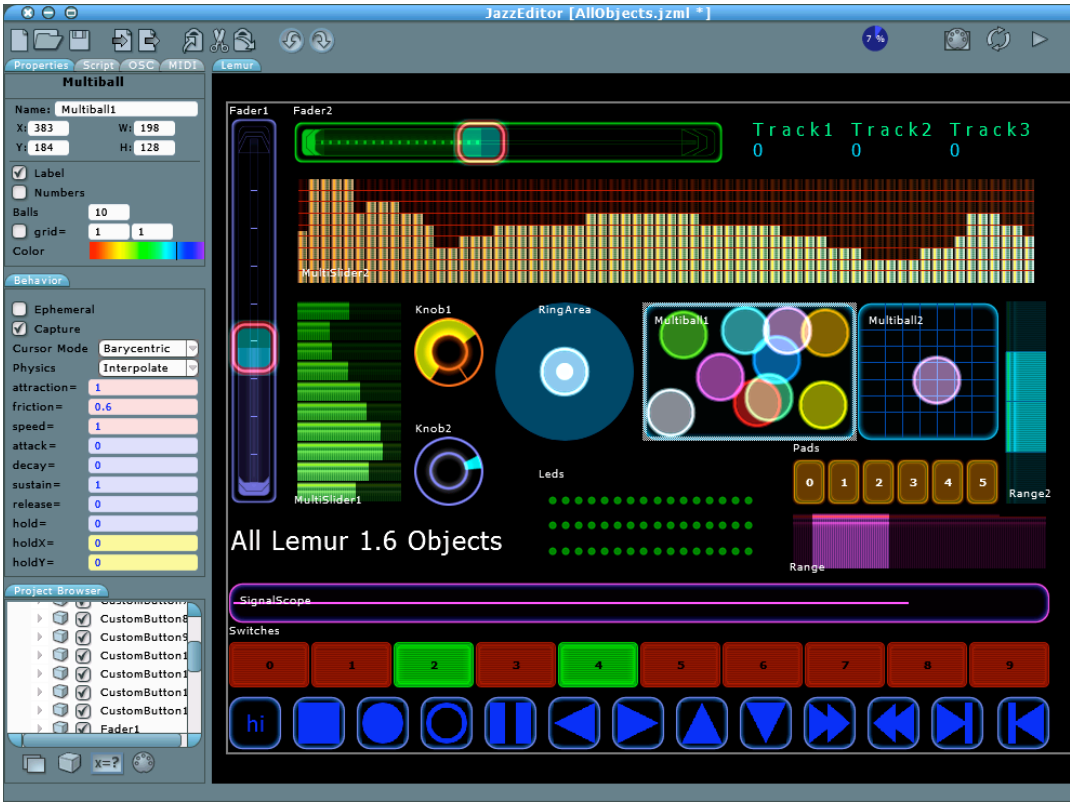


The MIDI panel presents all things you need for transforming the Variables of your Objects into MIDI messages and routing them to the MIDI Targets. This panel also serve for controlling Lemur Objects via MIDI.

3.1.9. Workspace Configuration

The layout of the JazzEditor Workspace can be customized to suit your preferences. Just grab a panel by its title bar, be it the Project Browser, the Object Panels or the Editing Area, and drag it to another spot on the Workspace. Everything can be rearranged.

The picture below shows a configuration that shows the complete Editing Area of your Lemur and access to the two Property Editors on a small, 1024x768 screen..

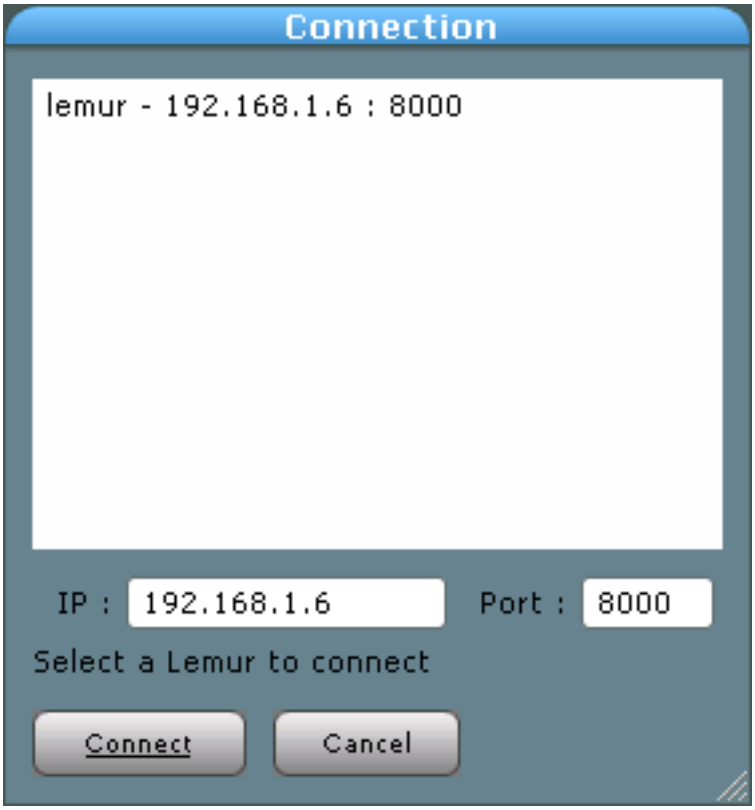


3.2. Interface building

Now that you know your way around the Workspace, it is time to get your feet wet and mess around with a little Project.

3.2.1. Connecting your Lemur

The first step naturally consists in opening the connection between JazzEditor and the Lemur. To do so, click on the **Connect** button on the toolbar, which should result in opening the following window :



If you have your network set up correctly (see the chapters about network configuration and the respective appendix), the connected Lemurs should be listed with their IP addresses and port numbers. You can select a Lemur on the list and click **Connect** to establish a connection. A double-click on the list entry does the same trick.

If you just connected your Lemur or had it switched on shortly before, wait a few seconds for the list to update.

Note : Be careful! If you have a Project residing on your Lemur and connect to it with a different Project open in the JazzEditor, the one on the Lemur will be overwritten. A connection always automatically transfers the currently opened Project to the Lemur.

The IP and Port fields below the list allow manual entry and remember the last setting of the Lemur connection. They also can be handy for connecting through a proxy.

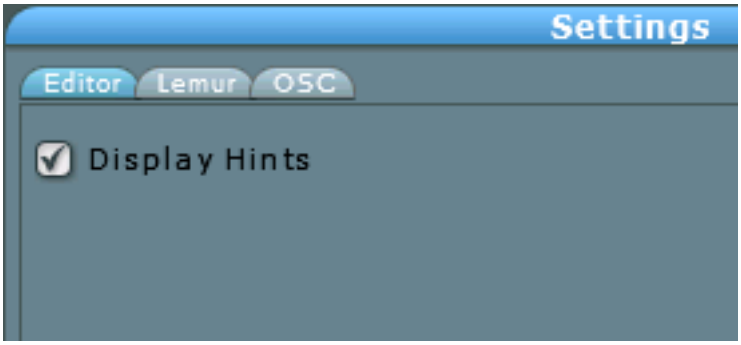
The current connection status can be seen from the displayed Connection icon. A **Play** button means there is no Lemur currently connected. If you connect to a Lemur the **Stop** button appears and a click leads to a disconnection of the Lemur.

3.2.2. *The Settings window*

3.2.2.1. Editor setting

The first panel of the Setting Windows deals with JazzEditor settings: you only have a checkbox for the display of mouse-over **hints**. When active, the

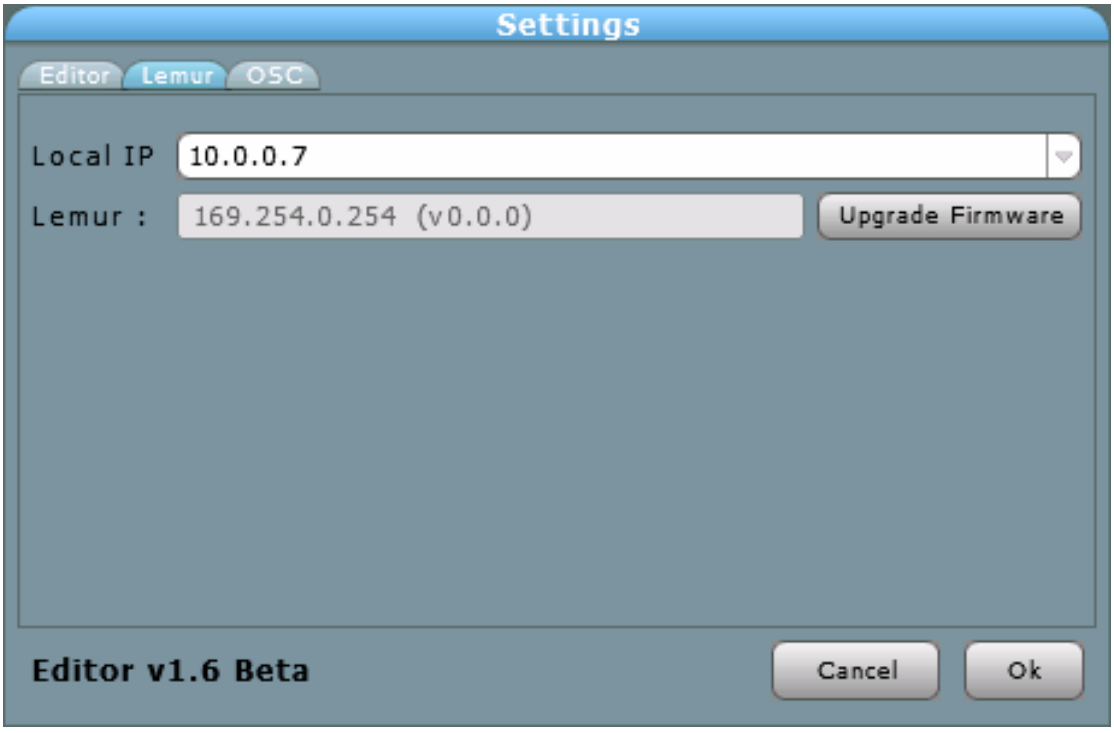
description of the interface item alongside with the eventual **keyboard shortcut** is displayed when you **hover with your mouse pointer** over the various interface elements.



3.2.2.2. Lemur Settings

On the second panel of the Settings window, called Lemur, you can choose which of the **Local IP** addresses of your computer is used by the JazzEditor for connecting to the Lemur.

The IP of the currently connected Lemur is also displayed.



3.2.2.3. OSC Settings

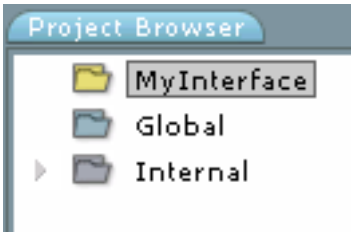
The OSC Settings are documented in chapter 3.3.

3.2.3. Creating an interface

Now that the Lemur and JazzEditor are connected together, let's create our very first interface. To do so, just click on the **New Interface** button, located below the project browser.

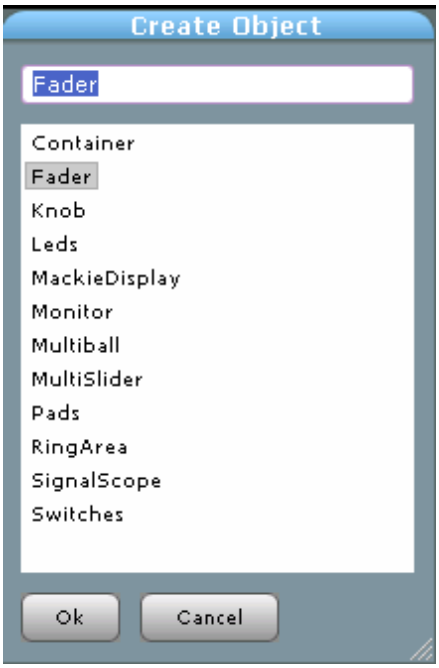
This opens a small dialog asking for the **name** of the new Interface. You can type in any name you like. Click OK and your new Interface is created.

Both the Editing Window and the Lemur screen suddenly turn black. There is nothing to worry about: we just created a blank interface. The yellow sit lemur you were familiar with is kept in a safety place – you will see it again next time you power on the device or you create a new project. You will also notice that the new interface appears on the project browser, as shown below.



3.2.4. Creating Objects

Although empty screens might be of a great interest from an aesthetical point of view, they happen to be rather boring and useless when they come to controlling software. To start filling this empty space, let's click on the **New Object** button, which is supposed to open the Object List Window, as shown below.



As its name may suggest, the object list window displays all available objects, sorted by alphabetical order. To create a new object, we just have to choose one among the list, to give it a name of our liking then to click the "Ok" button. For our first experience with interface building, I would suggest to start with the most common: the Fader.

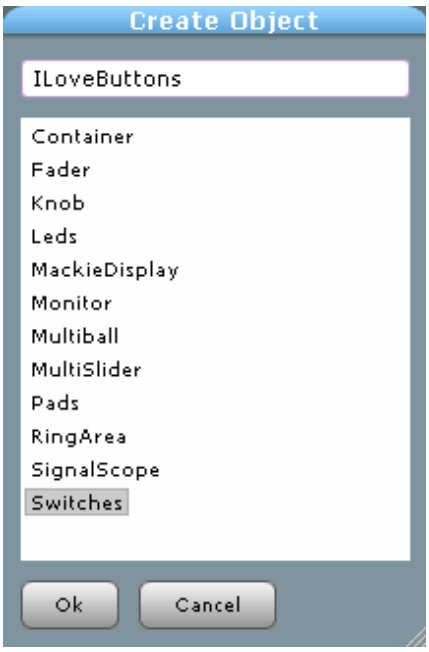
Once created, our newbie Fader appears on the upper left corner of both the editing window and the Lemur's screen. It's also referenced in the project browser.

As a reward of our effort, let's have some fun with the fader.

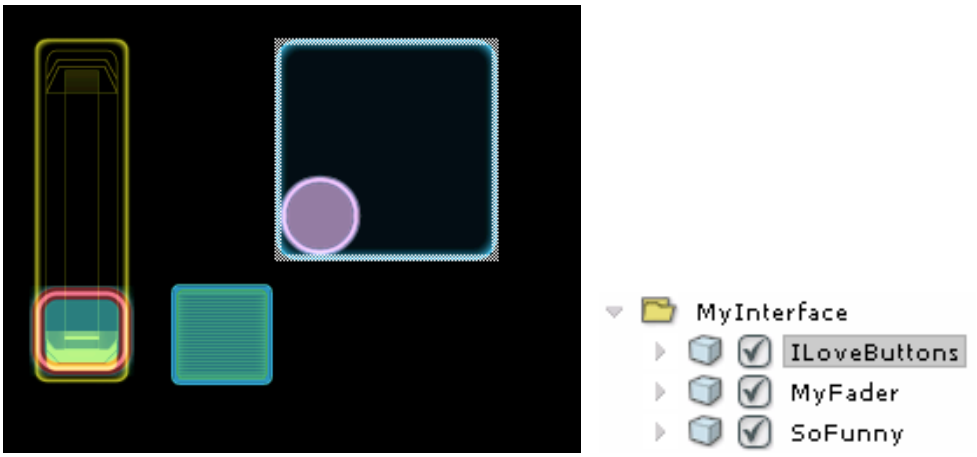
When you touch the Fader on the Lemur, you will notice that the actions are not mirrored in JazzEditor.

Click the Synchronization button to change this. Now everything you do on the Lemur will be reflected on your Computer. This helps if you have to save a Project in a defined state for future use.

Are you already bored with your lonely fader? That's no problem; let's provide it with a few fellow objects. By repeating the very same procedure described above, we're going to create a Switch and then a MultiBall.

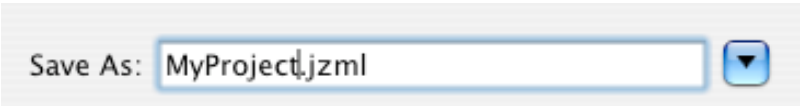


Not surprisingly, these will appear on both the editing window and the Lemur screen. And as good news never come alone, the two objects are also listed in the project browser, sorted in alphabetical order.



3.2.5. Saving your project

You might want to keep this work of art ready for future loading and editing. Just hit the **Save button** in the toolbar and give a name to your project.



When you do that, all the interfaces, objects and variables contained in your work session are saved in a single **.jzml** file.

All files the JazzEditor produces, be it Project files or Modules, reside on your hard disk.

The Project files have the extension **.jzml** and Modules have the file extension **.jzlib**.

Now, let's open our project again, by hitting the **Open button.** You might have noticed that the "Connect" Button has changed is status. That's a normal behaviour. When opening a new Project in JazzEditor (either by loading an existing one from disk or by creating an empty Project), the connection to the Lemur is dropped.

If you connect the Lemur again, the **Project** loaded in JazzEditor will **automatically be transferred** to the Lemur, overwriting the one on the Lemur.

There's currently no way to recover a Project residing on the Lemur. Make sure to save your work in the JazzEditor before closing the Project.

Once your Project is transferred to the Lemur, you can close the JazzEditor and use the Lemur to produce OSC data. It will happily connect to all defined OSC Targets.

If you use MIDI in your Project the JazzDaemon has to run, as it is needed to translate the OSC data from the Lemur into MIDI messages.

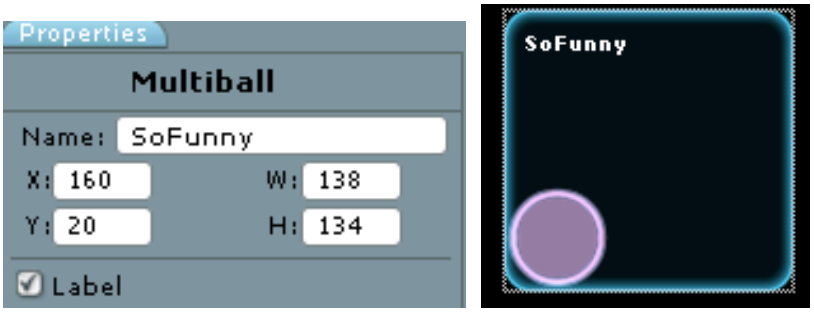
As a note for technical nerds, the Lemur and JazzEditor read projects as XML formatted files, leaving you endless possibilities for algorithmic interface generation, and other fancy geeky stuff. That's another story though.

3.2.6. Changing Object appearance

First we are going to **resize** the Fader to have some more travel for our fingers. **Grab the lower right corner** of the Fader and drag it down to make the Fader longer and maybe a bit wider, too. Note how the **Width** and **Height Parameters** in the Properties change as you do it, and vice versa: if you type a value in one of the dimension fields, the fader will move and resize accordingly.

Talking about the properties tab, what about changing the colour of our fader? Sure, a yellowish green can be nice, but what about a deep blue?

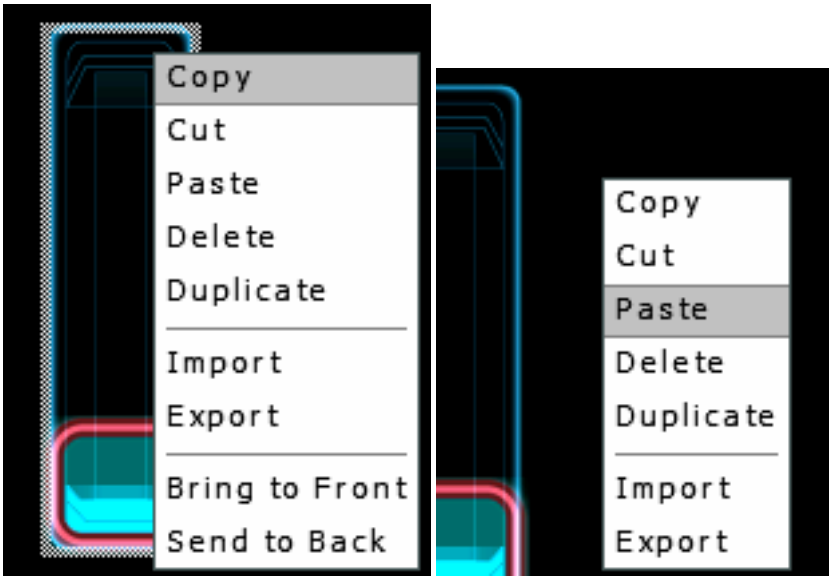
Sometimes it's a good idea to display the name of your Objects on the Interface. Click into the **Name** field of the MultiBall's General Properties and change it. Then check the **label checkbox**.



3.2.7. Basic Interface Work

3.2.7.1. Copy, Paste & Co

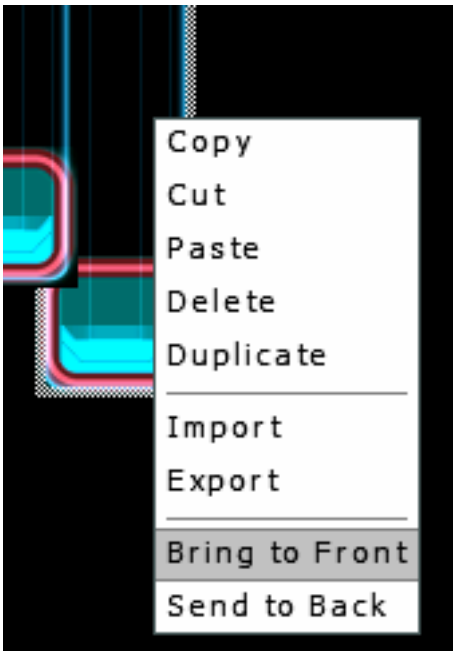
Now we want **two Faders**. We can do this with a simple copy and paste.



Right-click (Command-click for Mac) on the Fader and choose **Copy** from the **context menu**. Then right-click on the background of the Edit Area and choose **Paste** from the menu. Those commands are also available on the **Toolbar**. The Fader is duplicated. Grab it with your mouse and position it to the right of the first Fader.

Have a look at the General Properties of your Faders. The JazzEditor automatically named the new Fader "**MyFader2**" to avoid a **name conflict**. This is important, as the names are also used as addresses of the Objects for OSC mapping and variable accessing. As a consequence, there cannot be any identical names in any Lemur Project – except by protecting objects inside containers, which we'll cover later in this manual.

Of course you can also change the names by entering one of your choice into the name field of the Properties.

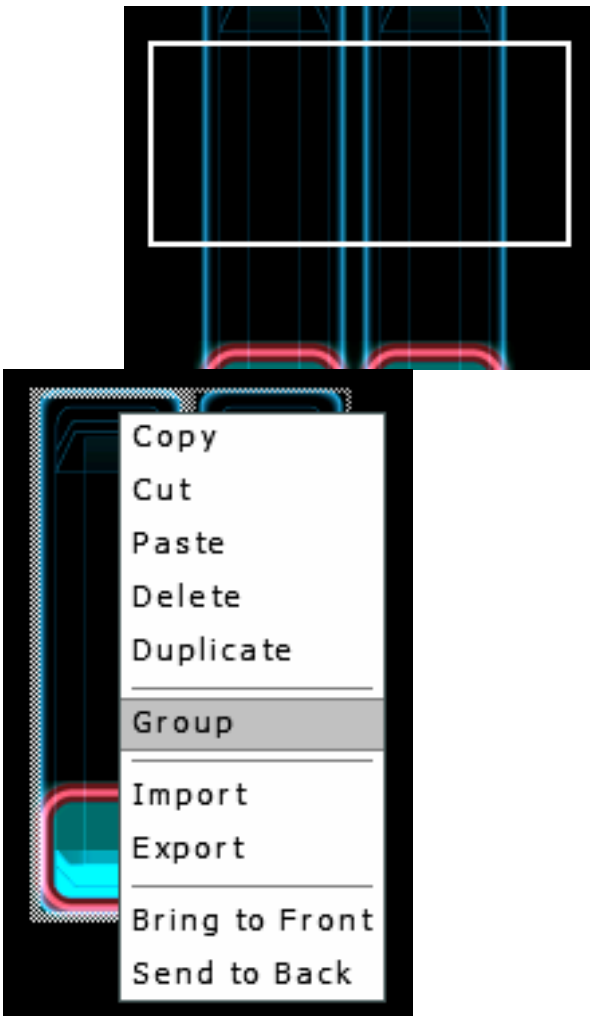


If you have several Objects lying on top of each other, the JazzEditor provides a way to control the layering order. A right-click on a selected Object

opens a menu. Choose **Bring to Front** to make the selected Object the topmost. **Send to Back** puts it to the bottom of the Object pile.

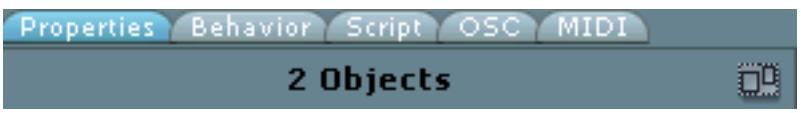
3.2.7.2. MMM

We now want to change the **colour** of the two Faders. As we already saw, we could do it by changing every Fader on its own. But it's tough to hit the exact same colour twice, so we select them both and change their colour to a same one.



Drag a **square** with your mouse that touches both Faders. Now they both are **selected** and you can change the properties for both Objects at the same time.

Objects can also be grouped permanently via the context menu by right-clicking (command-clicking for Macs) on the selected Objects and choosing **Group**.

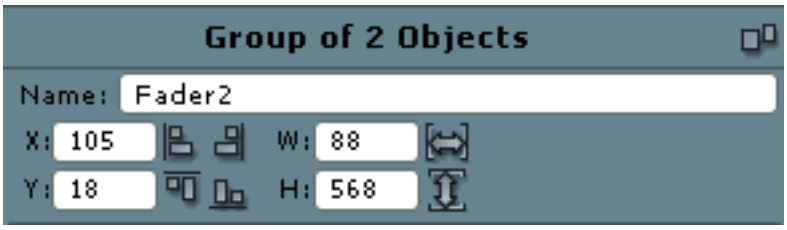


This group command is also available on the Properties panel, if several Objects are selected.

Do this for your Faders and every time you click on one of them both will be selected. Double-click on your new Group to display the General Properties panel. Note : since different kind of objects can grouped together, only the properties common to all objects are displayed. Now dial in a nice purple on the **color swatch**. Both Faders now have the exact same color.

If you want to ungroup the Faders, click on the **Ungroup** in the properties panel.

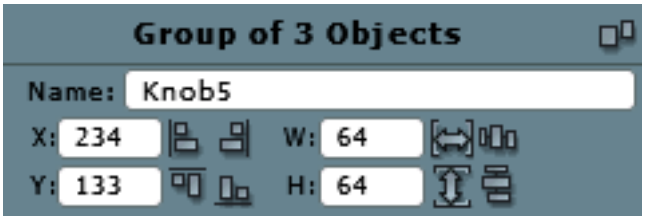
When you select several Objects or have them grouped, you'll notice additional commands appearing on the Properties Panel that deal with **layout** and **size** of the Objects..



The commands are:

- Align to Left, Align to Right
- Align to Top, Align to Bottom
- Make same Width
- Make same Height

They always refer to the Object you selected first in the group. These commands are of great help when it comes to tidying up your Interfaces. Experiment a bit with those to get the hang. And always remember that you have the **Undo** command to roll back changes.

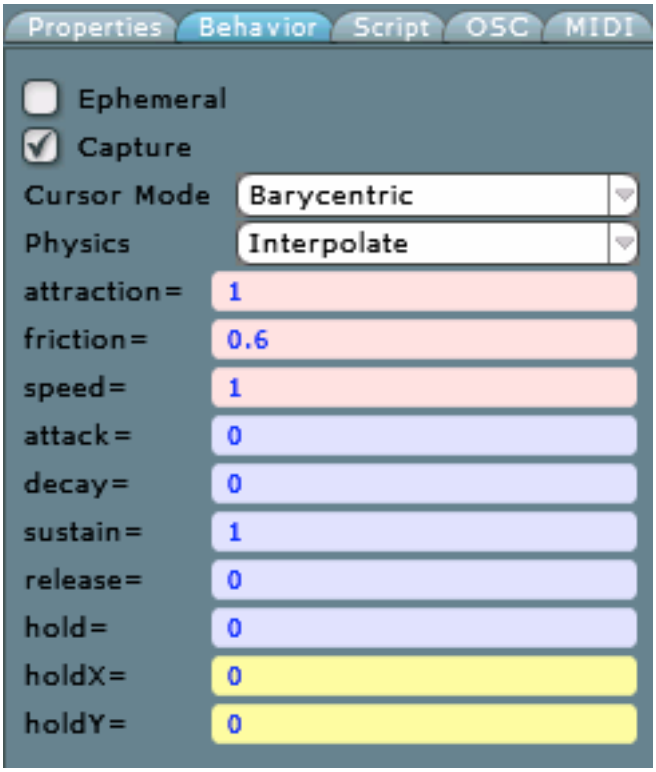


If you select three or more Objects (or have a group of three or more), two additional commands are available:

- Distribute horizontally evenly distributes the Objects on the screen. The left- and rightmost Objects are the boundaries for the line of Objects.
- Distribute vertically does the same, only on the vertical axis.

3.2.8. Configuring Object's behaviour

Now, let's glance at one of the most powerful feature of the Lemur's objects : their configurable behaviour. Select the MultiBall object with the mouse, and look at the Behavior panel. The following display will appear:



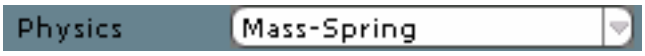
The **Physics menu** has three modes defining the behaviour of the balls. When Physics is set to **None**, the ball moves immediately to the position of your finger and stays there. It also follows your finger around as you move it on the touch screen, and immediately stops when you lift up your finger.

Make sure the Physics menu is set to Interpolate, which should be the default setting.

Touch the Lemur screen somewhere in the MultiBall, but not on top of a ball. The ball will not move to your finger immediately but according to the setting of the **Attraction** property. Let's modify Attraction to make the ball react more **slowly**. Enter a value of **0.1**.

Experiment again with the Ball's reaction. It should follow your finger slowly.

Now let's try the last **behavior** mode. Choose **Mass-Spring** from the **Physics** menu.



Change the Attraction value to 1.0 and the Friction to 0.2.

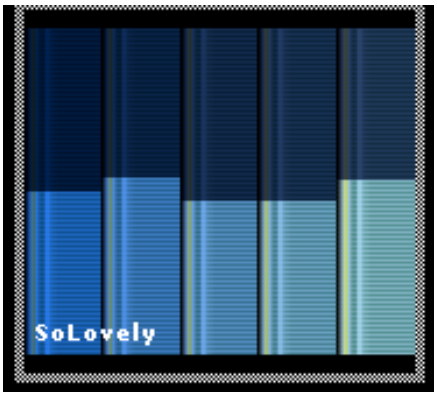
Try moving the ball now. It bounces off the "walls" and eventually slows down. The Mass-Spring mode is something like Interpolation, except that the ball has **friction** (or a lack of it), can bounce off of things, and in certain cases, the

ball may oscillate before coming to a complete rest. You can also control the speed of the ball via the **Speed** parameter.

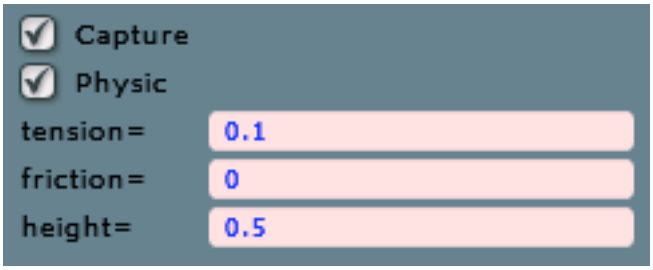
Physics parameters are nice to produce complex time-varying values with little effort. We will see later in this chapter that we can obtain even more enjoyable effects by controlling the Physics properties with other objects.

As another example we create a **MultiSlider** Object.

In the General **Properties** we enter **5** in the **Slider** field for having – you guessed it – 5 sliders. Also, we drag the MultiSlider to a free spot on the Interface and make it bigger so that we can interact easily with the individual sliders.

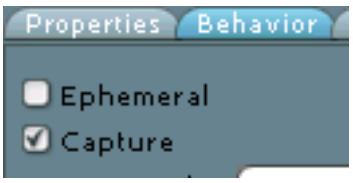


The default MultiSlider Object displays no physical behaviour. The sliders simply follow your finger and stay where you leave them.



Now change to the **Behavior** panel and activate the **Physic** checkbox. Also, change the **tension** to **0.1** and the **friction** to **0**. Now it's a completely different story. If you move the sliders they will continue to wiggle forever. They behave as if they are connected with springs and oscillate around a **centre value** that's defined by the **height** parameter.

Again, we are only scratching the surface here. Please have a look at the **Object Reference** for details about all Object Properties.

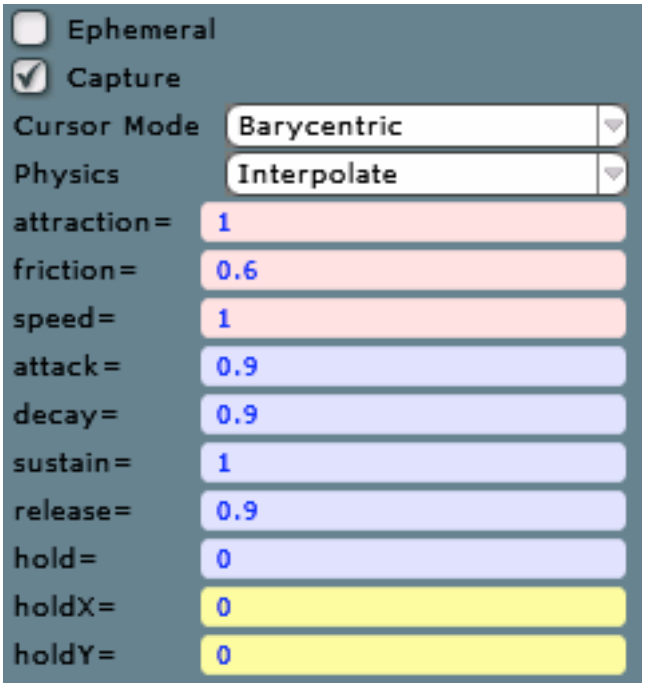


Let's come back to our MultiBall Object. The **Capture** parameter on the Behavior panel has great influence on the possible finger actions. When you are inside of the **Object's boundaries** the Capture mode doesn't make any difference. But if you move the ball and leave the borders of the Object, the control of the ball will stop when Capture is **off**. If Capture is **on**, the ball will continue being tied to your actions. No matter if you move your finger across other Objects and even if you switch to a different Interface, the ball will still cling to your finger and the finger has no effect on other Objects. Just experiment a bit with the two modes and you will quickly get the hang of it.

The MultiBall Object is a complex critter. It also sports an **ADSR+H** envelope for its brightness parameter that is accessible as a numerical value inside its z parameter. You might be familiar with envelopes from using synthesizers. They represent an easy way to produce defined parameter changes over time.

The MultiBall ADSR acts just like it's cousins from the synthesis world. When you touch the Lemur the **brightness** of the ball will be **faded in** according to the **Attack** value, **decay** down to the **sustain** level and fade out with the **release** time when you lift your finger from the surface.

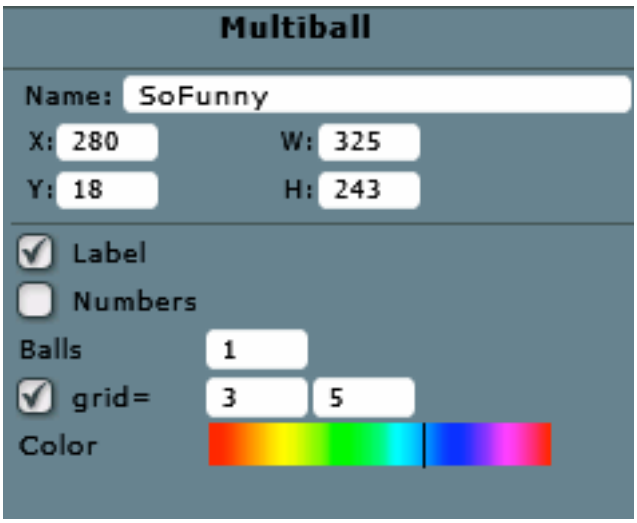
Objects supporting ADSR envelopes (MultiBall and Pads for the time being) also have a **hold** parameter. Its effect is similar to a sustain pedal, freezing the object's state as long as its value is 1. When set to 0, if has no effect. Interesting effects can be achieved when setting the hold parameter to an **expression depending on other objects' state.** We'll cover such methods later on.



The above picture is an example of brightness ADSR. Please note that the **Ephemeral** mode has to be active for the envelope to work. In Ephemeral mode the balls appear when you touch the surface and they vanish when you lift your finger.

Sometimes you don't want the **full resolution** of an Object because the target parameter in your software is **quantized** to only a **few values**. You can quantize the values sent via OSC by using variables and expressions, but the steps introduces by the quantization won't be reflected in the behaviour of the Object.

Enter the **Grid** parameter. You get it several kinds of objects. This parameter quantizes the movement of your Object into multiple steps. The number of steps can be chosen. Test it with the *So Funny* MultiBall object. Open its Properties and check the grid flag after setting its x-value to 3 and the y-value to 5.

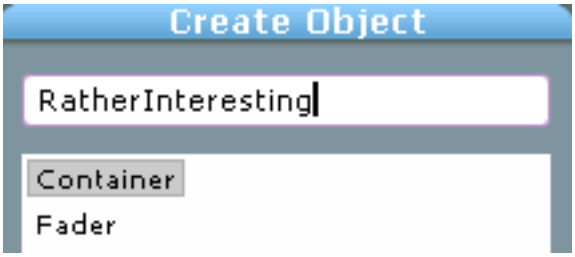


If you now have a look at the Object you see that the MultiBall area displays a grid and the Ball can only move to distinct points on every axis.

The values produced by the SoFunny.x and SoFunny.y variables are of course also quantized to produce only three steps on the x-axis (0.000, 0.5000, 1.000) and five steps on the y-axis (0, 0.25, 0.5, 0.75, 1).

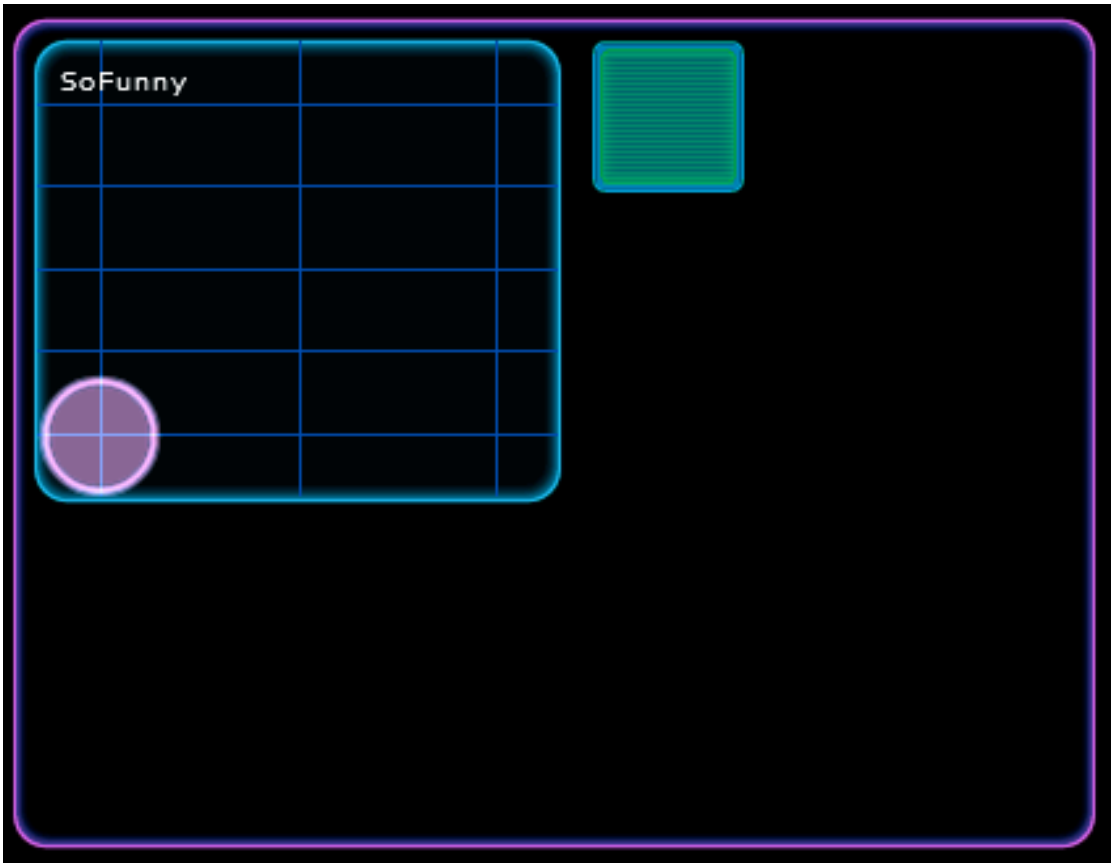
3.2.9. *Using Containers*

The next thing we want to do is put some of the Objects into a **Container** to separate them visually from our Faders. To do this, drag a **square** around the **MultiBall** and the **Switch** Object to select them. Now right-click (Ctrl-click on Mac) on the selection and choose **Cut** from the menu. The Objects disappear. They are not gone, though, but just got copied into the **clipboard** of the JazzEditor.

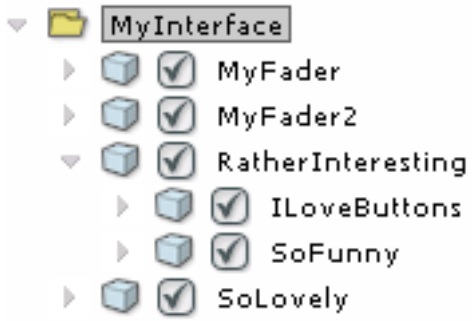


Click the **New Object** button and create a **Container** from the displayed list. Name it and position the Container to the right of the Faders and resize it to encompass the complete right part of the Interface.

Right-click (Command-click for Macs) the Container and choose **Paste** from the menu. This puts the two Objects we have cut out **into the Container**.

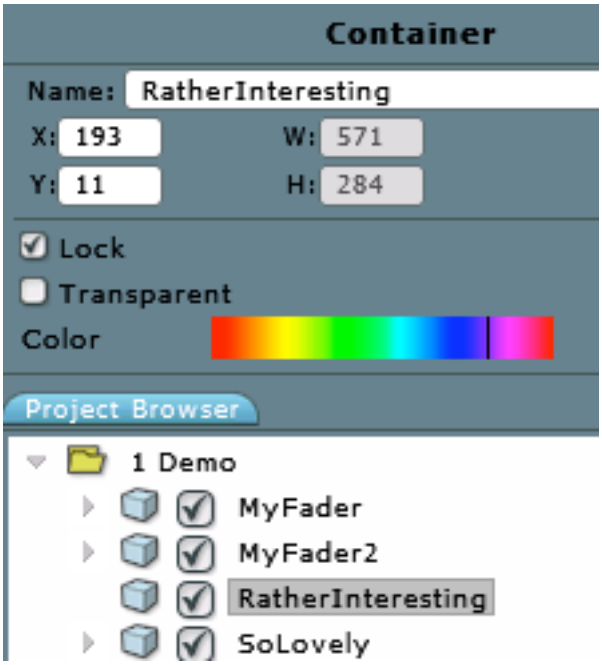


Now it's time to have a look at how all of this is mirrored in the Project Browser. The **Container** can be **opened** in the Project Browser by clicking on the **disclosure triangle** in front of the Container. You see that the two Objects in the Container are grouped on a lower level in the Project hierarchy.



Please note, that the use of Containers also has implications for the **Variable addresses** of Objects living inside of the Container. Indeed, within the Container, the local names of both objects are still the same. From the outside however, there are now known as *RatherInteresting.ILoveButtons* and *RatherInteresting.SoFunny* respectively. Do you feel a little bit lost? Well, that's normal at this point, but don't worry, we will come back to this later.

In larger Projects the Project Browser can quickly become quite crowded with Objects. This may lead to a lack of overview. You can make life a bit easier by using the Lock feature of Containers. Just try it with our *RatherInteresting* Container. Check the **Lock** flag in the Properties and you will see the MultiBall and the Switch vanish from the list of Objects.



A locked Container doesn't show its contents in the Project Browser. This of course also means you can't edit the Objects inside of it. Simply unlock it again, if you need access to the contents.

3.2.10. *Import and Export of Modules*

Let's suppose now that you would like to reuse later some portions of the interface in another project. For that purpose, the JazzEditor provides the quite useful **Import Library** and **Export Selection** commands of the Toolbar for creation and integration of **Modules**. Any imported Module is incorporated into the currently opened Interface and you can select any group of Objects for export as a Module to the file system. Please note that you can also import and export Interface elements via the context menu of the Project Browser (right-click/control-click on the Editing Area).

The file extension for Modules is **.jzlib**.

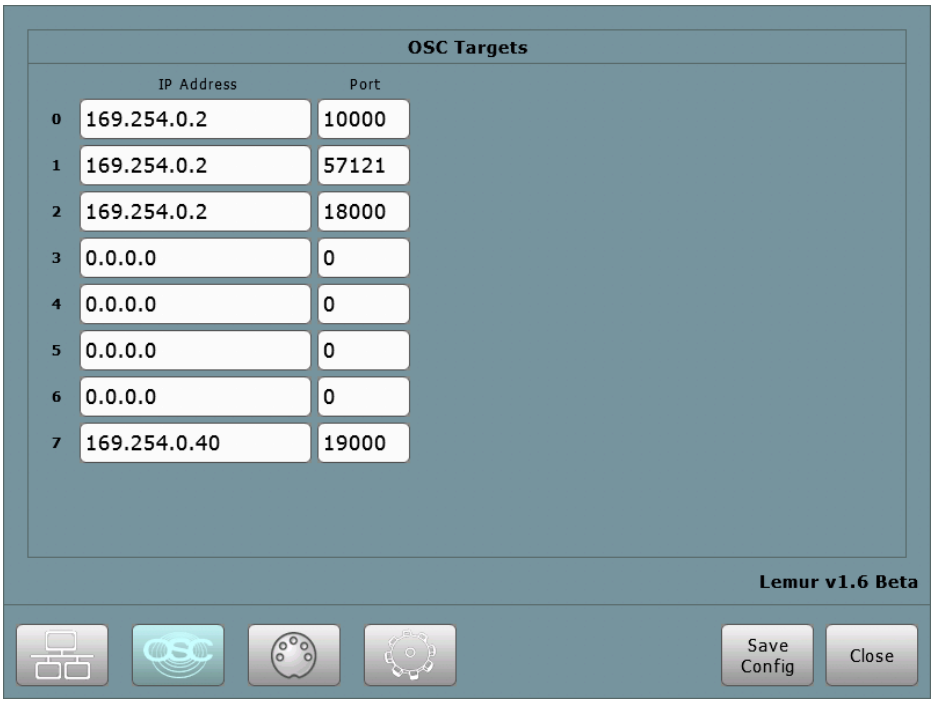
There are some caveats concerning **Object hierarchy** when it comes to exporting parts of your Projects. Make sure that you do not destroy any **dependencies** in terms of variables and expressions. It is good practice to **encapsulate** the exported Module in a Container and have the variables and functions that are important for the Module's functionality created **locally** to that Container. If you define them globally, they will not get exported and the functionality of the Module will be broken

3.3. *MIDI and OSC Target Setup*

3.3.1. *OSC Settings*

The **Open Sound Control** protocol specifies the transmission of messages between two devices. Rather than attaching specific meanings to these

messages - as it is done with MIDI - Open Sound Control allows you to define your own system of messages. With the Lemur for example, the names of objects you create and their "path" in the Project hierarchy constitute their default "address" for OSC messages.



The second screen of the Lemur's Settings deals with the eight **OSC Targets**. You can reach it via the **Settings button** on the Lemur. The Targets are also mirrored on the **OSC** Settings in the **JazzEditor,** if a Lemur is connected. You can use both screens to set up your OSC Targets.

If there is no connection, the JazzEditor states this and does not show any OSC settings.

OSC Targets can be software applications on your local computer, somewhere on the network, or OSC-enabled hardware connected to the network. They are called Targets because they are what the Lemur targets its messages at. Always remember that OSC is **bi-directional** and the OSC Targets can also control your Lemur's Objects.

Just fill in the IP addresses of the Targets you want to communicate with. If it's a piece of software running on the same computer as the **JazzEditor**, click the My IP button on the OSC panel of the JazzEditor. This will automatically fill in the IP of the currently used interface.

On the **Lemur** you have to mark the IP Address with your finger and fill in the new one via the appearing number pad. Do the same for the port number.

The **Port** setting is dependent on the corresponding setting in the Target. Ports are something like channel selectors that allow multiple applications to share the same IP address without conflicts. The choice of port number is determined by what software you are using:

- **Port 8001 and 8002 should not be used**, because the JazzEditor and JazzDaemon use those ports to establish connections with the Lemur. If you try to use another application while the JazzEditor is open, the application will not be able to access ports 8001=8002. Similarly, if you launch the JazzEditor while another application is using ports 8001-8002, it will be unable to connect to the Lemur.
- For **Max/MSP**, the port number can be anything. 8000 is the typical value used.
- For **Reaktor 5** the port number must be higher than 10000.
- For other OSC-compatible software or hardware, consult the Target's documentation for details on port settings.

Close closes the Settings window and saves the OSC settings into the Lemur's flash memory. Wether you set up your OSC targets from the JazzEditor or from the Lemur, they will always be saved to memory and available at each reboot.

Once you have loaded a Project from the JazzEditor onto the Lemur (which is automatically done on connect) or open a Project from the Lemur's internal flash memory, the OSC messages are handled by the Lemur. This means you can close the JazzEditor and let the Lemur communicate with OSC Targets.

3.3.2. MIDI Settings

MIDI Targets can either be set up on the **Lemur** itself or via the **JazzDaemon** running on a computer. For MIDI work there has to be a JazzDaemon running on the computer as the MIDI data is not generated on the Lemur itself: the Daemon works as a bridge, translating between MIDI and OSC protocols in real time.

The Daemon automatically scans all available **MIDI ports** on the computer. These may be physical hardware ports or virtual ports like offered by MIDI Yoke or Maple on WinXP, or IAC Busses on OSX(see 3.3.2.3 and 3.3.2.4 for details). The detected ports are then made available to the Lemurs on the network.

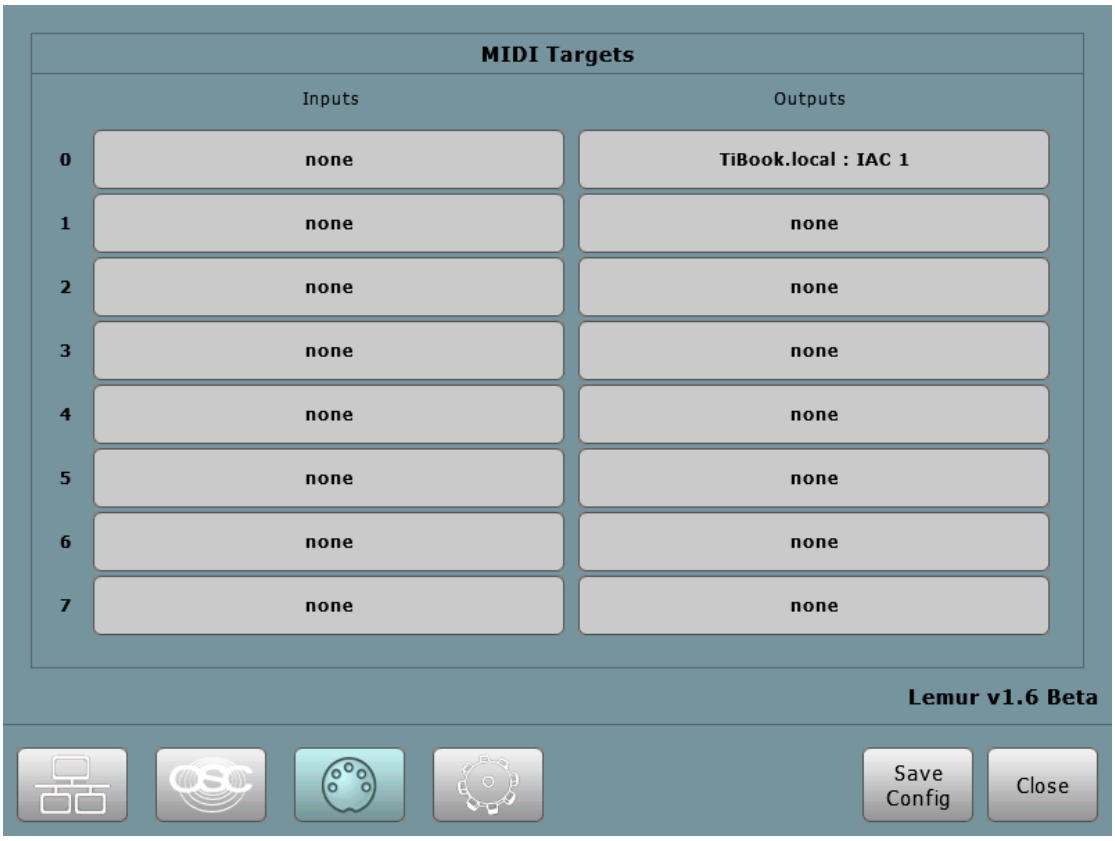
It doesn't end there. You can also have **several computers** on the same network all running MIDI Daemons and you will see all available MIDI ports of all computers on the MIDI Setup page of your Lemur. This means maximum flexibility in terms of MIDI topology. Regardless of the complexity of your MIDI network, you can reach every single device via your Lemur, as long as it is connected to a computer on the same network as the Lemur.

There are some thoughts to be invested about being able to make MIDI connections on the Lemur and making them on the Daemon. The two methods lead to different behaviour when starting the Lemur or the Daemon, respectively:

- The **connections created from the Lemur** can be saved in its internalf Flash from the **Save Settings** button. The Lemur will then automatically attempt to reconnect to the Daemons at boot. If the connections can't be re-established (due to a missing Daemon on the computer), the entries on the Lemur's MIDI settings are greyed out. At the moment an awaited Daemon appears on the Network, the Lemur makes the corresponding connection. If you always connect your Lemur to the same computer(s), saving the MIDI connections on the Lemur is highly encouraged : there will be no setup time afterwards.

- The **connections created from a Daemon** are saved in its preferences. They will be automatically remade when the Daemon is launched. If the connections can't be re-established (because the Lemur is connected), the entries on the Daemon's MIDI settings are greyed out. At the moment an awaited Lemur appears on the network, the Daemon makes the corresponding connection. If you need to have several MIDI setups depending on which computer the Lemur is hooked to, you should create the connections from the Daemons : this way, whenever you plug your Lemur to any PC, the relevant connections are made.

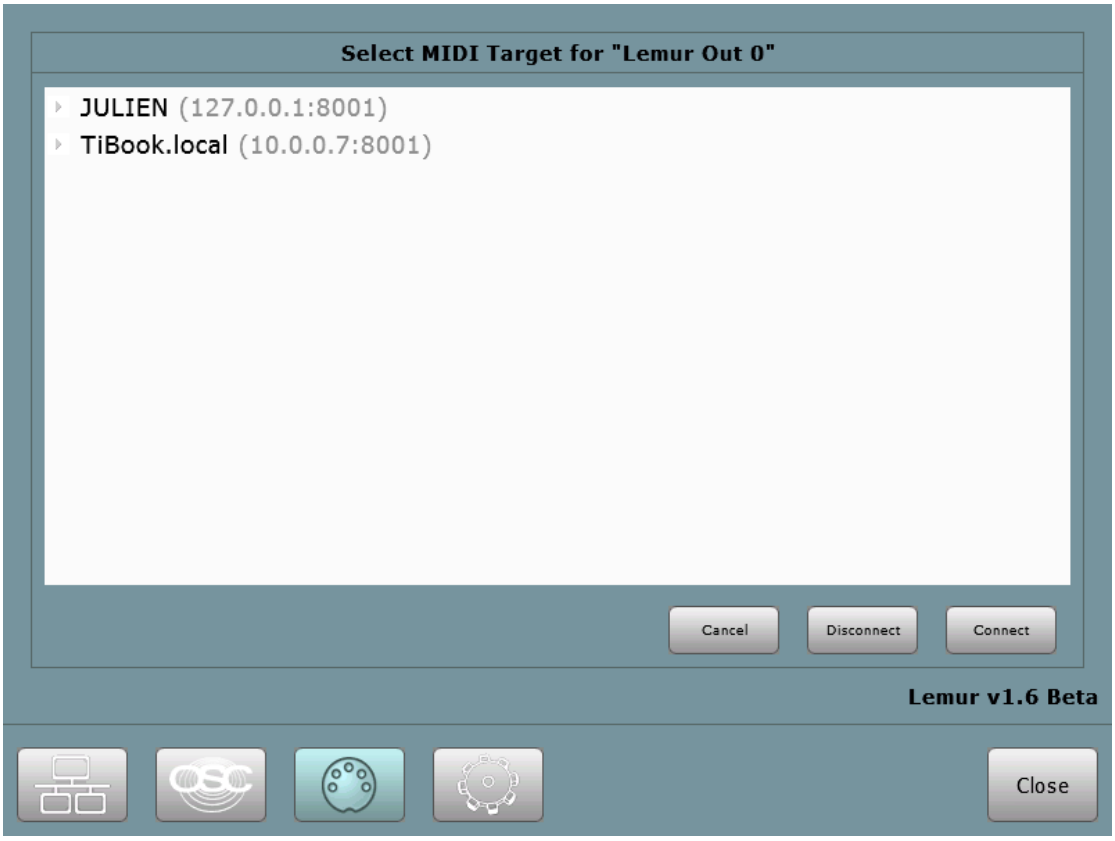
3.3.2.1. Lemur MIDI Settings



The **MIDI Settings** screen on the Lemur displays the currently defined **MIDI Targets**. There can be up to eight of them. The MIDI communication is bi-directional, meaning you can choose a **MIDI input port** (incoming to the Lemur) and a **MIDI output port** (outgoing from the Lemur). Which ports are present on the menu depends on the ports presented by the MIDI **Daemons** currently running on the network. Please have a look at chapter 3.3.2 for further detail.

A MIDI Daemon has to run on the computer for working with MIDI as the MIDI data is not generated on the Lemur itself.

Touch the **Input** or **Output** of a Target to be changed on the MIDI screen of the Lemur. You get a **list** of currently available Daemons on the network.



Touch the **small triangle** in front of the Daemons to open the **list of available ports** on the respective computer. Just touch a port and touch **Connect** to make the connection. You can also touch **Disconnect** if you chose an existing connection and it will be disconnected. **Cancel** closes the list without changing anything.

Touch **Apply** to save the changes you made into the Lemur's flash memory. The settings done on the Lemur will be remembered the next time the Lemur wakes up.

Close closes the Settings saving any changes you made to the internal memory of the Lemur.

3.3.2.2. MIDI Daemon Settings

MIDI Targets are defined with the **MIDI port browser** of the **MIDI Daemon**. Please read chapter 2.3 for details about starting the Daemon.

The MIDI port browser is opened via a **menu** you can reach by clicking on the Daemon icon in the System Tray (for Windows XP) or the Menu Bar (for MacOS X), respectively.

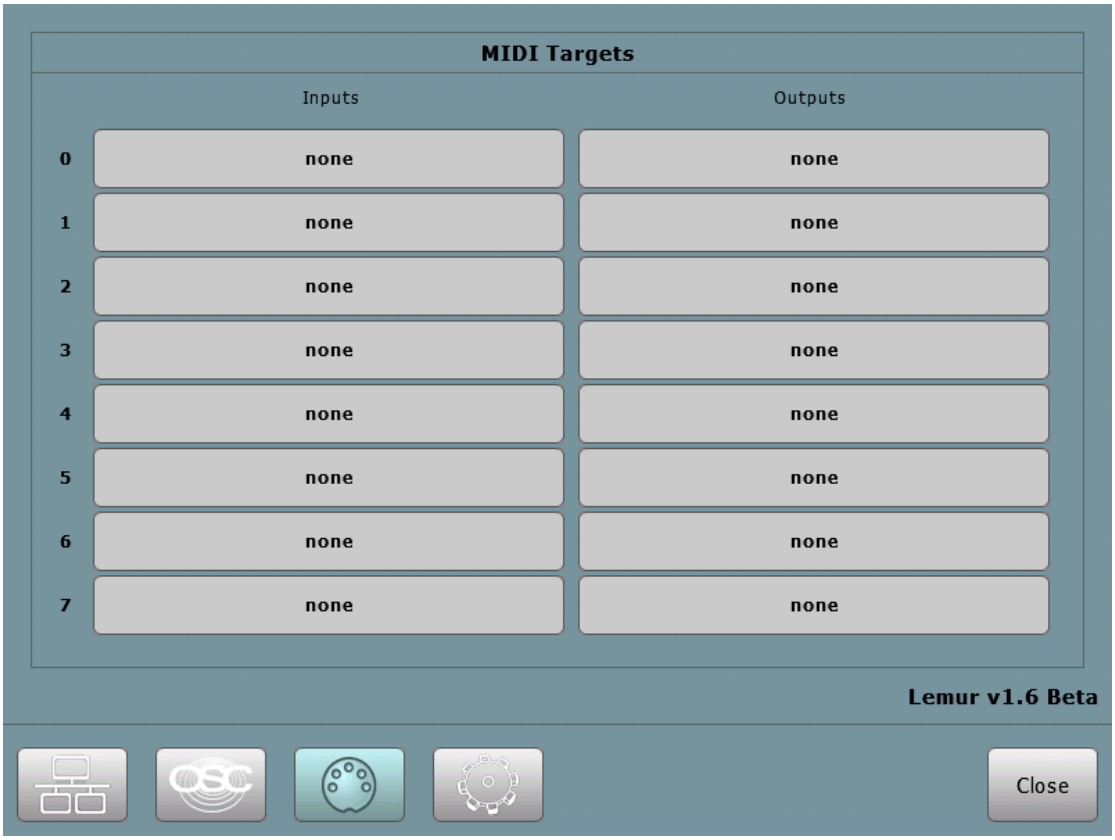
Setup MIDI Daemon opens the MIDI port browser. On the left side of the browser the available **MIDI input ports** on the system are listed. The right part shows the available **MIDI output ports**.

3.3.2.3. MIDI Setup Example

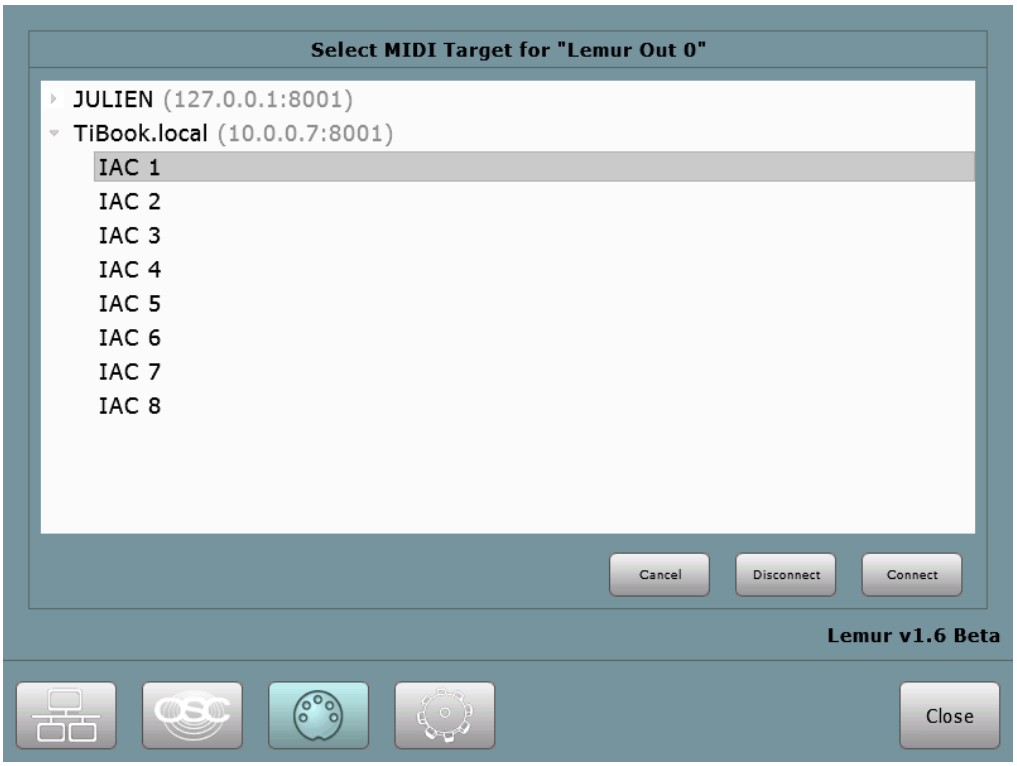
Let's do a small **walkthrough** showing the steps to get a MIDI connection up and running:

- Start the Daemon by going to the JazzMutant start menu folder and clicking MIDI Daemon.

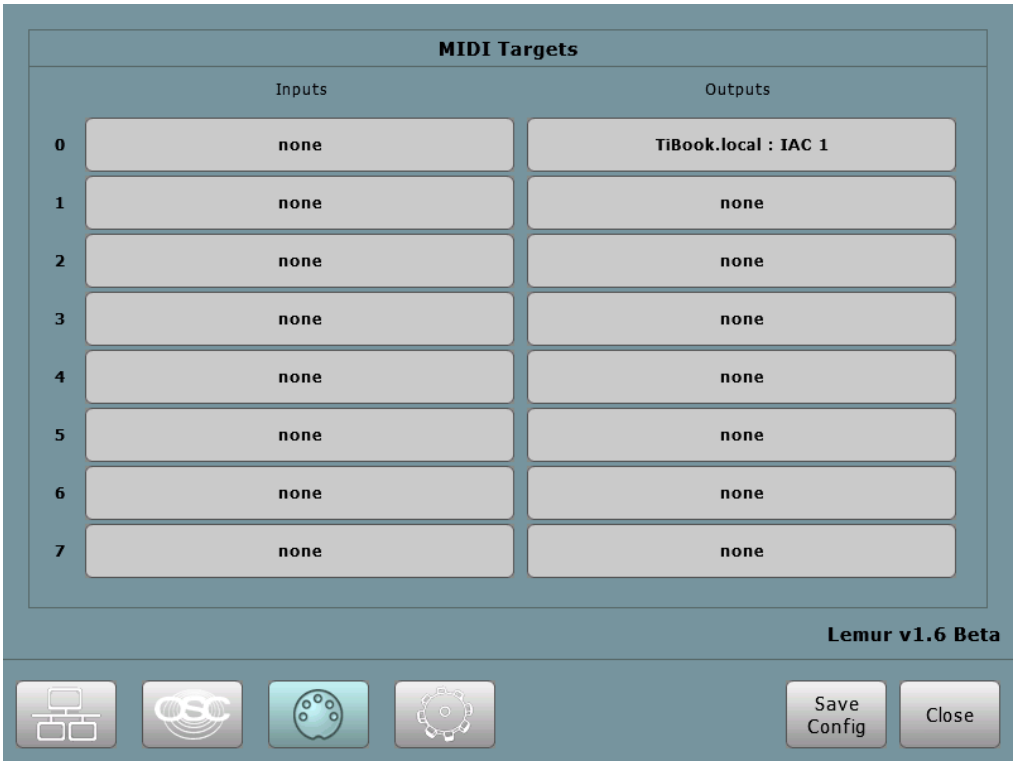
- Switch on your Lemur.
- Push the Settings button on your Lemur (the first button) and touch the MIDI button on the screen.



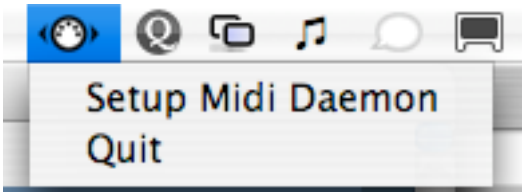
- Now touch the **Output button** for the **MIDI Target 0** and see the **list** of Daemons detected by the Lemur.



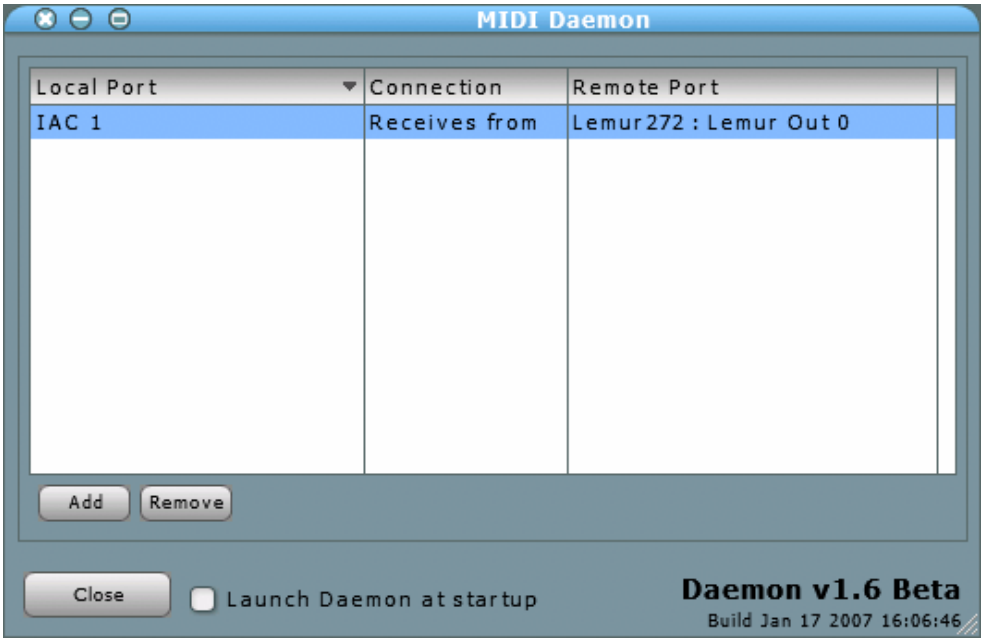
- Touch the small triangle in front of the computer name to drop down the ports list. Choose one of the **MIDI ports** (see below for details) and touch the Connect button.
- You should now see the selected port appearing in the Targets list.



- Now click on the **Daemon's** icon on your computer and choose Setup MIDI Daemon



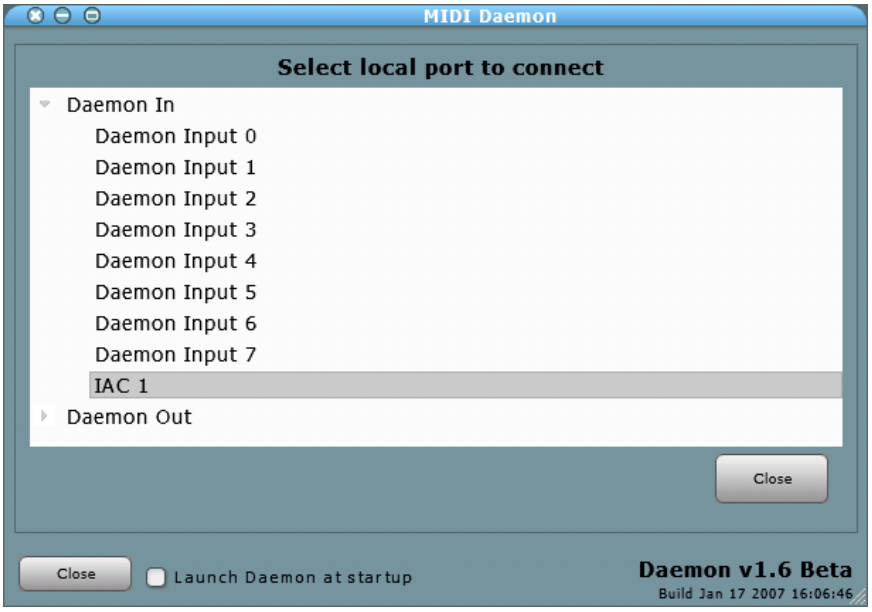
- The list for current MIDI Connections appear. It shows the connection that has just been done from the Lemur.



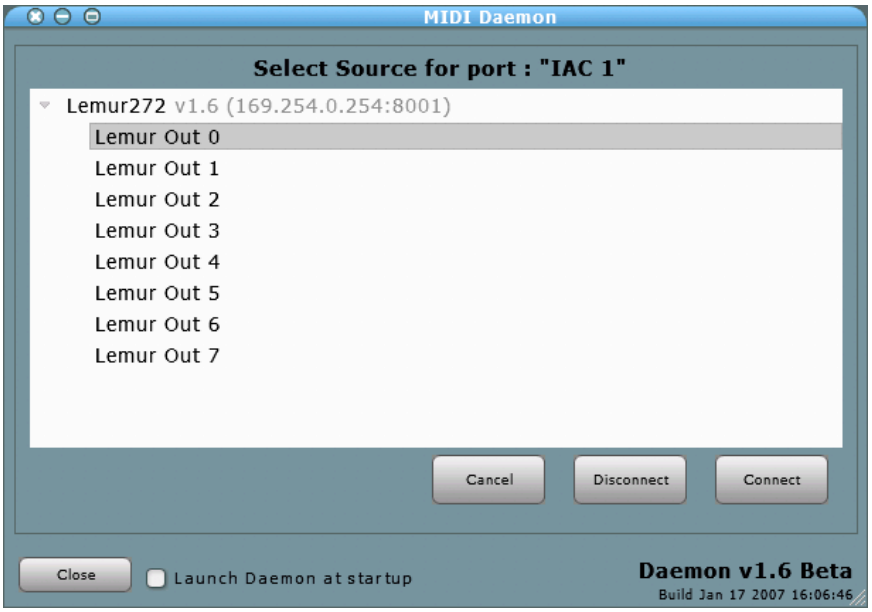
- You can now check if the **MIDI connection** works. Connect an **application** to the virtual MIDI port. Create a Fader and send some controller data to Target 0 or take a Pad Object and send some Note On messages.
- **Turn off** your Lemur
- **Turn it back on** and have a look at the MIDI Settings on the Lemur and at the Settings of the MIDI Daemon. Everything is still there and works as before.

The connection of a Lemur target to MIDI port can also be done the other way around.

- Click the **Add** button in the JazzDaemon window. You are now presented with all the MIDI ports of the machine. You can choose any In or Out port to connect to a Lemur. (Note that Mac Users will see there the Daemon Input and Daemon Output virtual ports that are automatically created by the JazzDaemon at launch). Any port in the **Daemon In** section can be connected to the output of a Lemur. Any port in the **Daemon Out** section can be connected to the input of a Lemur.



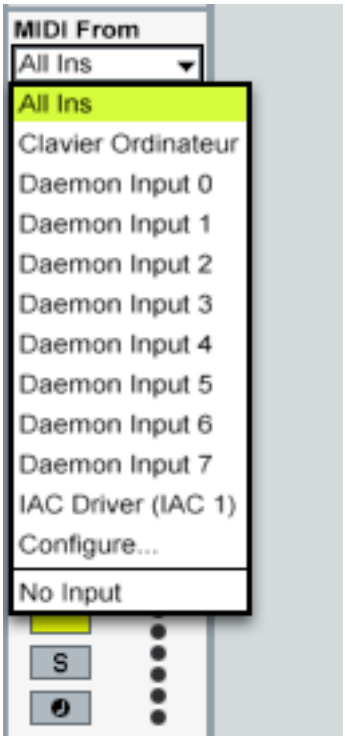
• Double click on the local port you want to connect. Here, we want to receive all MIDI data coming from the Lemur's target zero on our virtual port named **IAC 1**. You can now browse the network and look for a Lemur to connect to. Choose **Lemur Out 0** as a source, and double click the item, or hit the **Connect** button.



• Congratulations, you have successfully created a connection from the Daemon ! Now if you quit and relaunch the Daemon, or reboot your computer, it will automatically redo the connection from **IAC 1** to **Lemur Out 0**. Use this technique if you often swap computers connected to your Lemur, and need a unique configuration for each of those.

3.3.2.4. Virtual MIDI Ports on Mac

On the Mac, the JazzDaemon automatically creates 8 virtual Inputs and 8 virtual Outputs for you to link with Lemurs. They are conveniently named **Daemon Input** and **Daemon Output**. Once you've connected them to the Ins and Outs of Lemurs, you can use them in any MIDI application on your Mac.

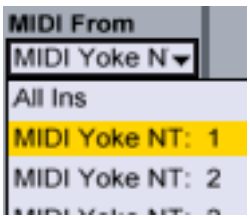


3.3.2.5. Virtual MIDI Ports on Windows™ XP

On Windows, you'll need a third-party software virtual MIDI interface, such as **LoopBe** or **MidiYoke**. Both drivers are **freeware** and easily downloadable from the internet. The virtual MIDI drivers offer multiple MIDI ports that you can use to create multiple connections from the Lemur to your host.

In the software application you want to control with the Lemur, choose the virtual MIDI port to which the Lemur is connected. As you can also have bi-directional communication via MIDI - i.e. the MIDI-enabled software can also control Lemur Objects - use a second virtual port to make the reverse connection to the Lemur.

Again, we will take Ableton Live as an example for the Target side. If you want to route MIDI data from the Lemur to a live track, choose the appropriate virtual MIDI port from the MIDI input menu in Live.



3.4. Use your Interface

Now that we have set up our Interfaces and know how to modify the Objects it is time to put all this to use. You already know how to set up MIDI and OSC Targets by using the **Lemur Settings** from the Toolbar (see Chapter 3.2). We assume in this section that you have your Targets sorted out.

For every Variable of your various Objects you can decide to which Target its value is transmitted and from which Target its variables and parameters are controlled. You can choose **MIDI** and **OSC** Targets **independently**. This ensures maximum flexibility in terms of routing values across your software and hardware setup.

3.4.1. The send mode

As we said before, a project can be comprised of a great number of variables. Only a few of them are useful when it comes to controlling your application - most are only meant to change objects properties. Also, as we will see later, there might be at least two reasons for not sending the output value of a particular object. One would be if you were using an Object merely to display information coming from your computer. The MultiBall object could represent the state of something happening in different application.

The other reason could be that you transmit the data from the Object to your computer via a **Variable** containing a mathematical expression. In this case the original Variable (e.g. the x) is only referenced by the Expression in another Variable. This is often used to scale values into a more appropriate range. We'll learn more about this technique later.

Since it's not worth overloading the network with useless data, we provided an option that let you decide whether or not a variable shall be sent.

Let's have a look at the project browser window.

The **checkboxes** next to My Fader and x are what we're interested in. the checkbox in front of the Fader must be checked for enabling any data transmission from this Object. Also, the checkbox next to the **x** must be checked in order for the Lemur to send fader positional data when you touch the Object.

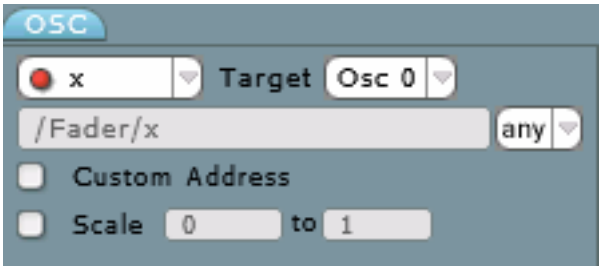
We want the Fader to transmit its position to our setup, so we should **check My Fader** as well as **x**.

While checkboxes are enabled by default when creating objects and variables, remember to uncheck the variables you don't want to be sent out to reduce network overload.

3.4.2. Setting up OSC mappings

If you are unsure about **basic OSC setup**, please have a look at chapter 3.3.1

Make sure our Fader is selected and have a look at the **OSC panel**.

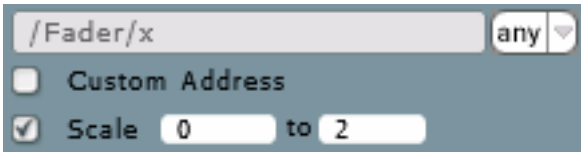


On the left you see the **Variables** menu and to the right the **Targets** menu. We want to transmit the Fader's **x Variable** via OSC. If you have a look at the **Variables menu** of the OSC panel, you see that it lists all available Variables of the Fader: Choose the x. From the OSC Targets menu, choose the Target that you have set up in the general OSC Settings.

If you now move your Fader it will send the value of **x** to the OSC address **/Fader/x**. Use whatever OSC software you have to check it.

If you want to change the OSC address use the **Custom Address** flag and type in whatever address you need. This may be useful in case of naming conflicts between Lemur Projects and projects running on the Target side.

Default parameters of objects range between 0 and 1 (such as the x variable of a Fader). However, you can choose to scale their OSC output to your own range. Enable the **Scale checkbox** and enter other integer boundaries for the output.



The communication with OSC Targets is **bi-directional.** You define the pairing for both **incoming and outgoing** OSC data with the various parameters of your Objects on the OSC panel. That's why you also find Variables on the menu that don't output values: they can be remote controlled from the displayed OSC address.

Main object's parameters react to incoming OSC according to their OSC range. Sending **/Fader/x 1.0** to the Lemur will bring the Fader's cap to the top if the defaut range is used. If you've customized the range to 0..2 as explained above, the Fader's cap will instead go midway, since 1.0 is thye center of the 0..2 range.

Note that OSC scaling is only enabled to object's main parameters, and not to user-created expressions, since those have undefined range by essence.

You might already have noticed the different colors for the Variables in the menu. By the way, the above menu is from a MultiBall Object.

- **Red** stands for the Object's main parameters (x for a Fader; x, y, z for a MultiBall, etc.)
- **Orange** depicts a properties/behaviour parameter (like friction, attraction, value, etc.)
- **Green** entries are Expressions created by the user.

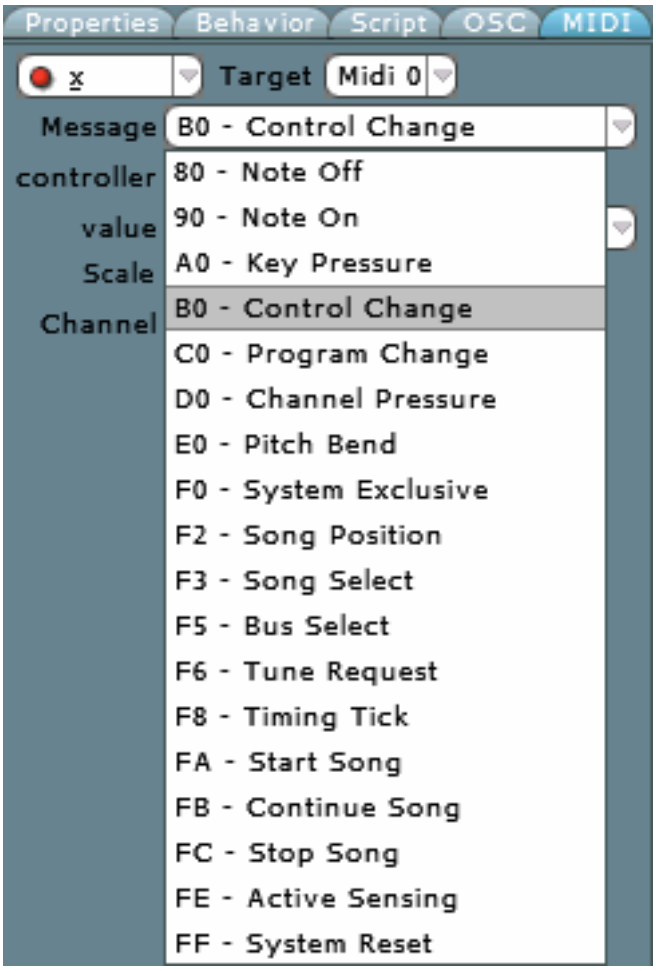
Have a look at Chapter 3.3.4 for more information about the **Trigger Mode** menu. It allows for choosing when and how often the messages are actually transmitted (triggered).

3.4.3. Setting up MIDI messages

3.4.3.1. Using the MIDI window

On the top of the MIDI panel, you'll find two menus. The left one is the **Variables menu** and the right one is for choosing one of the **MIDI Targets** defined in the MIDI Settings.

Choose any of the MIDI messages from the **Message Menu**. You find a great variety ranging from Note messages over Controllers to System Realtime messages, like Song Start/Stop or Active Sensing. They are preceded by their MIDI status byte.



Depending on the message type you get various parameters to tweak.

Please have a look at the **MIDI Message reference** appendix for details.

The communication with MIDI Targets is **bi-directional**. You define the pairing for both **incoming and outgoing** MIDI data with the various parameters of your Objects on the MIDI panel. That's why you also find Variables on the menu that do not output values.

You might already have noticed the different colors for the Variables in the menu.

- **Red** stands for the Object's main parameters (x for a Fader; x, y, z for a MultiBall, etc.)
- **Orange** depicts a properties/behaviour parameter (like friction, attraction, value, etc.)
- **Green** entries are Expressions created by the user.

Have a look at Chapter 3.3.2 for more information about the **Trigger Mode** menu. It allows for choosing when and how often the messages are actually transmitted (triggered).

3.4.3.2. Simple MIDI Mapping

In order to generate MIDI messages from the objects you touch on the Lemur, you have to establish a **mapping** between the 0-1 values transmitted by the Lemur and MIDI values, which are typically integers (whole numbers) between 0 and 127. This is done on a per Variable basis on the MIDI panel of the Object. We'll start with a simple example.

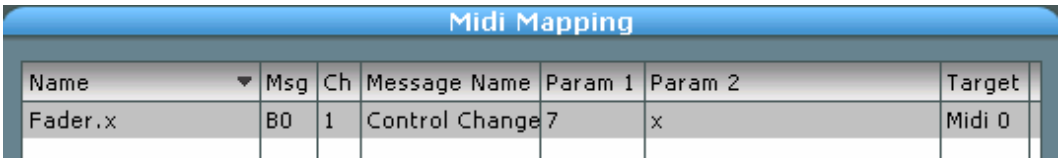
- Select a Fader in your project. To assign a MIDI message to this fader, click the **MIDI panel** of the Fader.

- Change the settings of the dialog to those shown above (**Message** is Control Change, controller number is 7, and value is x). The Scale fields should be set from 0 to 127. Set the MIDI channel according to the channel of the desired Target, which is

chosen via the Target Menu to the upper left. These settings configure the Fader to transmit MIDI controller 7 (volume) with the fader's 0-1 floating-point values mapped to 0-127.

- Verify the MIDI assignment by clicking on the **MIDI Map** icon at the top of the JazzEditor window.

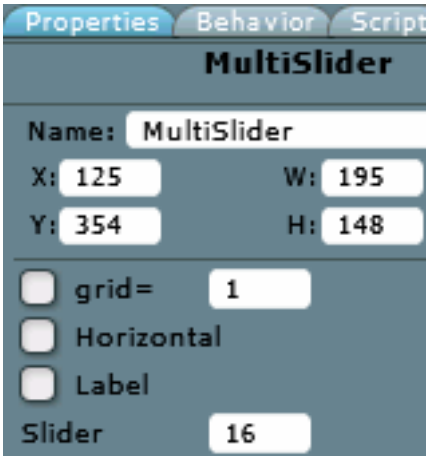
The MIDI Mapping window will appear. It displays the assignment we just configured.



3.4.3.3. Multiple Controller Mappings

In the next example, we'll use a **MultiSlider** object and automatically assign different control change messages for each slider.

- Create a MultiSlider object. Using the **General Properties** tab, set the number of sliders to **16**.



- Click the **MIDI panel** of the MultiSlider.

- On the MIDI panel choose B0 **Control Change** and MIDI 0 as Target for the x Variable.
- Type controller number **10** into the first **controller** field. The second controller field is automatically set to **25** to extrapolate for the 16 sliders.

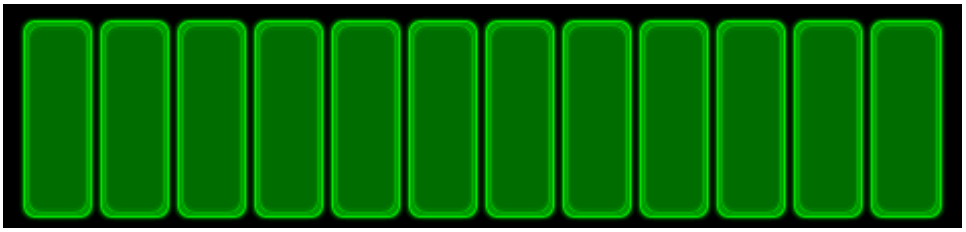
- Click the MIDI Map icon to open the MIDI Map window.

Since the MultiSlider has sixteen values instead of just one, the JazzEditor automatically assigns each slider to successive MIDI controller values. The same thing happens for other multi-value Lemur variables (such as MultiBall's' parameters).

3.4.3.4. Mapping MIDI Note Messages

The following example uses the same automatic mapping feature to create a chromatic one-octave MIDI "pad" on your Lemur that will send MIDI note messages.

- Create a **Pads** object. Give it **12 columns** and **one row**.



- Create a new **MIDI assignment**. Choose the **x** Variable and **Note On** as the MIDI message type.
- Set the first pitch field to **60** and the second will **automatically be expanded** to **71** (reflecting your 12 Pads). Your MIDI assignment should resemble the one shown below.

- Click the **MIDI Map** icon to look at the MIDI Mapping window. You see the 12 Pads nicely laid out working as a MIDI keyboard.

Name	Msg	Ch	Message Name	Param 1	Param 2	Target
Pads.x	90	1	Note On	60	x	Midi 0
Pads.x	90	1	Note On	61	x	Midi 0
Pads.x	90	1	Note On	62	x	Midi 0
Pads.x	90	1	Note On	63	x	Midi 0
Pads.x	90	1	Note On	64	x	Midi 0
Pads.x	90	1	Note On	65	x	Midi 0
Pads.x	90	1	Note On	66	x	Midi 0
Pads.x	90	1	Note On	67	x	Midi 0
Pads.x	90	1	Note On	68	x	Midi 0
Pads.x	90	1	Note On	69	x	Midi 0
Pads.x	90	1	Note On	70	x	Midi 0
Pads.x	90	1	Note On	71	x	Midi 0

You can use more than 12 pads to create a bigger keyboard if you wish.

3.4.3.5. Using the Midi Mapping window

The table consists of seven columns:

- The **Name** column shows the name of the Variables used.
- The **Msg** (Message) column displays the numbers of the different MIDI messages used for the individual mappings.
- In the **Ch** (Channel) column you find the associated MIDI channel for the individual mappings.
- The **Message Name** column writes out the full name of the different MIDI messages.
- The two **Parameter** columns show the MIDI parameters associated with the chosen message type, i.e. the CC number and the CC value for a MIDI controller message.
- The **Target** column shows which of the MIDI Targets that are defined on the Settings window have been chosen for the respective mapping.

The **Refresh** button updates the list of MIDI mappings and you can close the window by hitting the **Close** button or the escape key on your computer keyboard.

3.4.4. Trigger Modes

Common to the OSC and MIDI parameters that can change over time is the **Trigger Mode Menu**:

The **Trigger Mode** is set for every parameter individually via a dedicated menu and it decides when a message should be transmitted.

Let's look at the trigger mode for one parameter of a MIDI message (let's say, the velocity of a note on message, or the value of a control change)

any: The message is sent whenever the parameter changes

up: The message is sent each time the parameter rises from 0 into the positive value range

down: The message is sent each time the parameter reaches 0 from the positive value range

up and down: The message is sent each time the parameter reaches 0 OR rises from 0.

+: The message is sent each time the parameter increases above its previous value

-: The message is sent each time the parameter decreases below its previous value

For Custom MIDI Messages (see Chapter 3.5.5 for an explanation) there is an additional item on the Trigger Mode Menu.

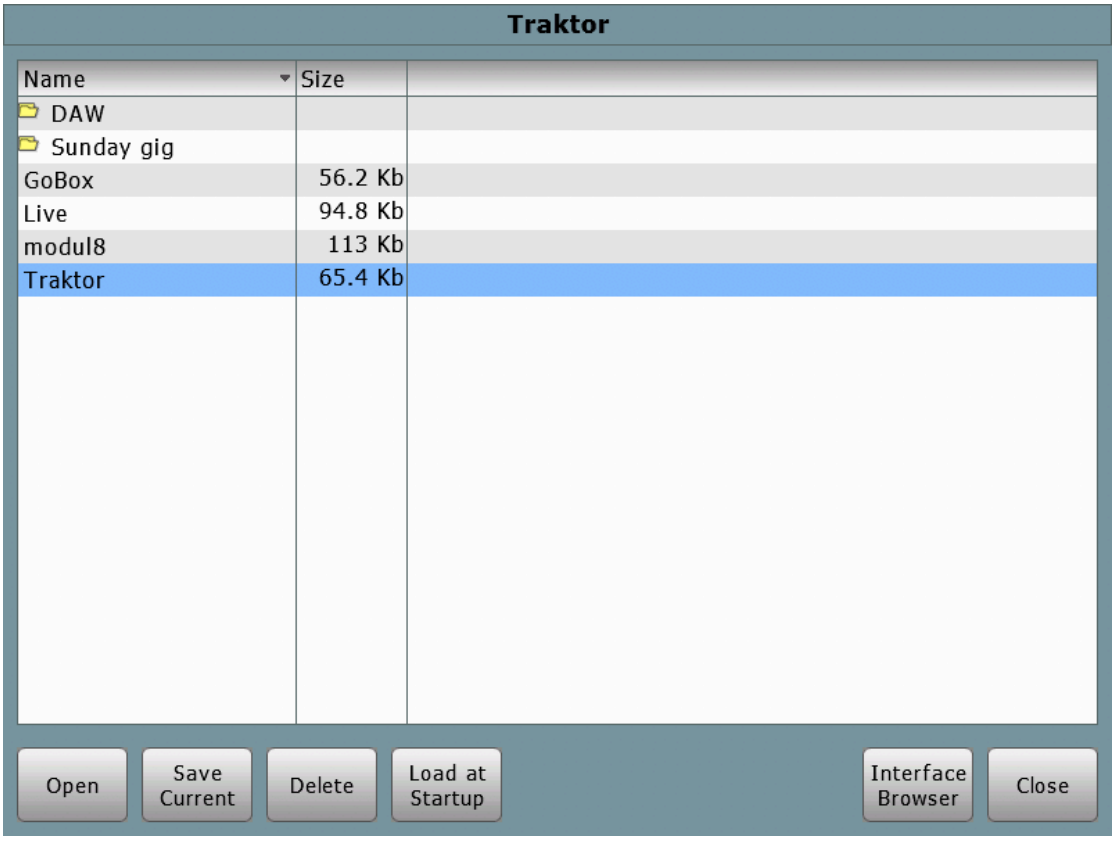
The first entry ("**None**") means that a change in the attached value or expression will not trigger a transmission of the MIDI message. It might, however, still get triggered by values or expressions associated with other parameters of the Custom MIDI message that don't have their Trigger Mode set to None.

3.5. Advanced features

3.5.1. The Lemur's memory for Projects

The Lemur has a **memory** for the **Projects** you throw at it. You just have to tell it to remember them. This frees you from always having to open the JazzEditor when you just want to work with your finished Interface.

Push the second button on top of the Lemur and hit the Project Browser button to open the **Project browser**. It displays a **list of Projects** currently stored in the Lemur's flash memory.



We want to save the important Interface we have built, so we touch the **Save Current** command. You see the Project appear on the list. You can select the Project ((leading to a blue tag) and touch **Delete** to remove it from memory and **Open** to load the selected Project into the main memory.

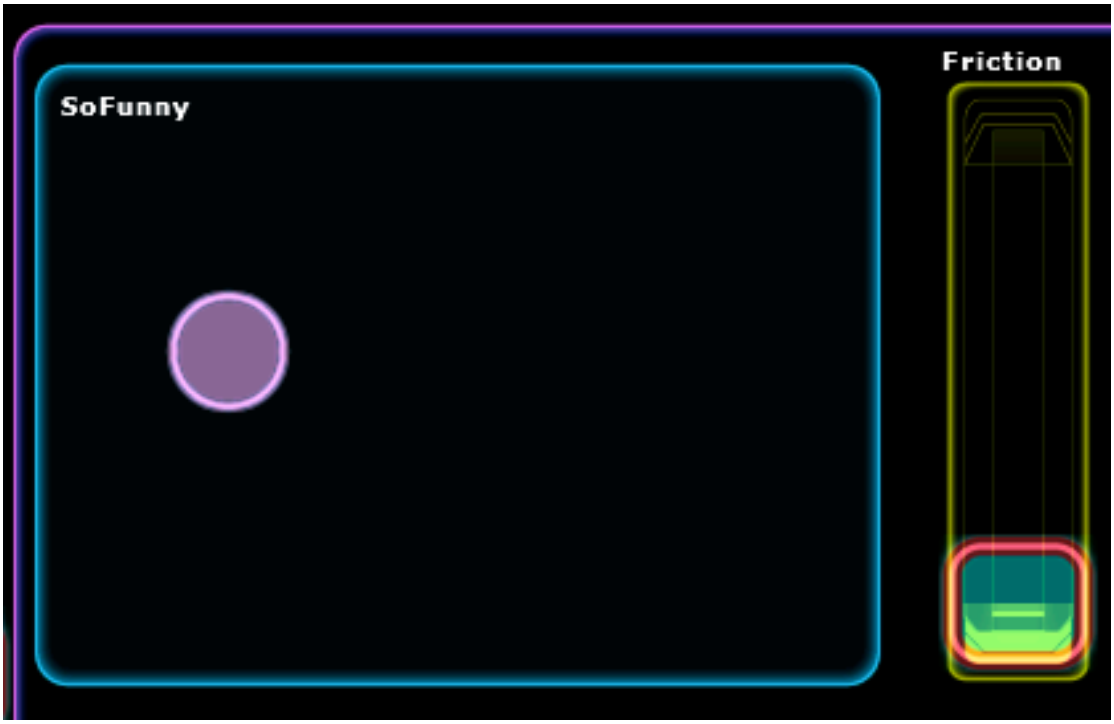
If you want the Lemur to open a specific Project when the Lemur is switched on, select the Project on the list and activate the **Load at Startup** flag. The Startup Project is tagged in red. Touch Load at Startup again to remove the tag.

You have a maximum of 2048 kb for storing Project data on the unit.

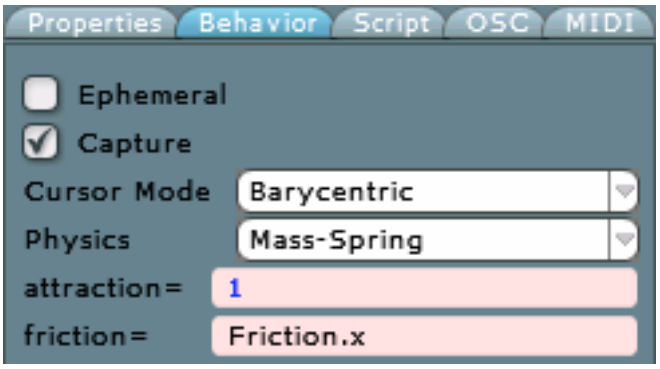
Use the Close button to shut down the Project browser and get back to working on the JazzEditor. You can also hit the **Interface Browser** button to go back to the Interface thumbnails.

3.5.2. Control your Objects with your Objects

It's easy to control the behaviour of your Objects via other Objects on the Interface. Let's add an additional **Fader** to our original Container. Call it **Friction**.



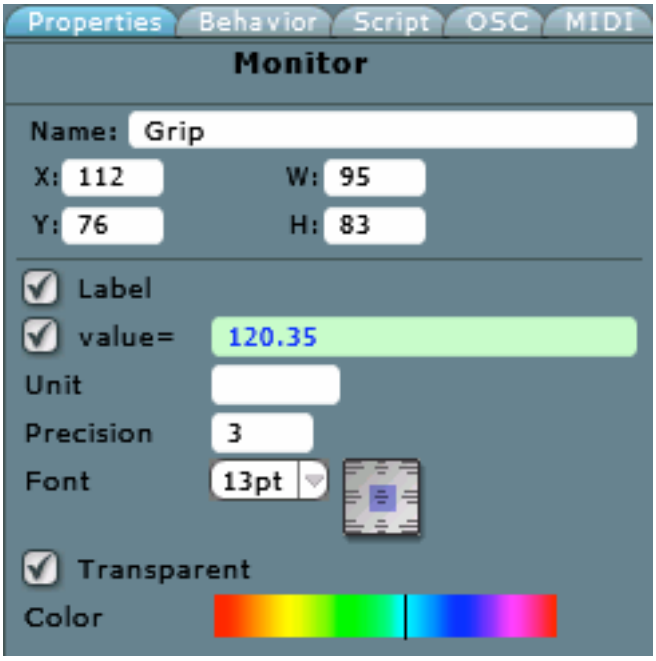
Now switch to the **Behavior** panel of the **MultiBall** Object and change the Physics parameter to **Mass-Spring**. Please **de-activate the grid** as it is of no use in Mass-Spring mode. Also, enter **Friction.x** as the value for **friction**.



Now you can control the friction of the MultiBall via the new Fader. It's as simple as that. Use any value of any of the Objects dwelling on your Lemur Interface to control any of the parameters. This can lead to very interactive Interfaces.

If you have **sub-objects** you can address their behaviour properties **individually** by using a **vector** or list instead of a singleton. If you have **three balls** in a MultiBall Object and use a vector with **three components** to denote, say, friction, all balls will get their individual friction value. Of course this also works with attraction and speed and **any other parameter** of objects with multiple **sub-objects**.

You could have the Fader display the chosen friction value. But sometimes you want to have the value displayed at a different place on your interface. You can even display values on top of Objects. Let's do that now. Create a **Monitor** Object inside of our **Container** and position it right in the middle of the MultiBall area.



Name the Object **Grip**, check the **Label**, **Transparent** and **value** flags. Fill in **Friction.x** for value because we want to display the friction parameter set via our Fader.

Now set the Font size of the Monitor to **13pt** and choose a nice, contrasting **color**. It should look similar to this:

If you move the Friction Fader, its value is displayed via the Grip Monitor. You can still use the **MultiBall Object**, as if the Monitor wouldn't exist, because the Monitor is transparent to touch. Don't mix this feature up with the **Transparent flag** of the Monitor which is only making the background of the Monitor invisible. You can always "reach through" a Monitor Object, even if its Transparent flag is unchecked. This can come handy if you want to hide Objects on your Interface but still use them with your fingers.

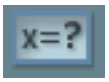
3.5.3. Making your own Variable

Now, let's say you have that special Reaktor patch with a custom parameter having an extra wide range. You want fine-grained control with a Fader in the lower range but you also need the possibility to control it over the complete range. How would you do that?

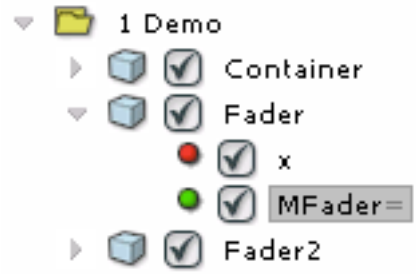
Simply take a second Fader and use it as a **multiplier** for the Value of the first one. This is best done via the creation of a **local Variable** that gets transmitted via OSC and MIDI.

Go back to our two originals faders from the beginning of this chapter, and select the first one.

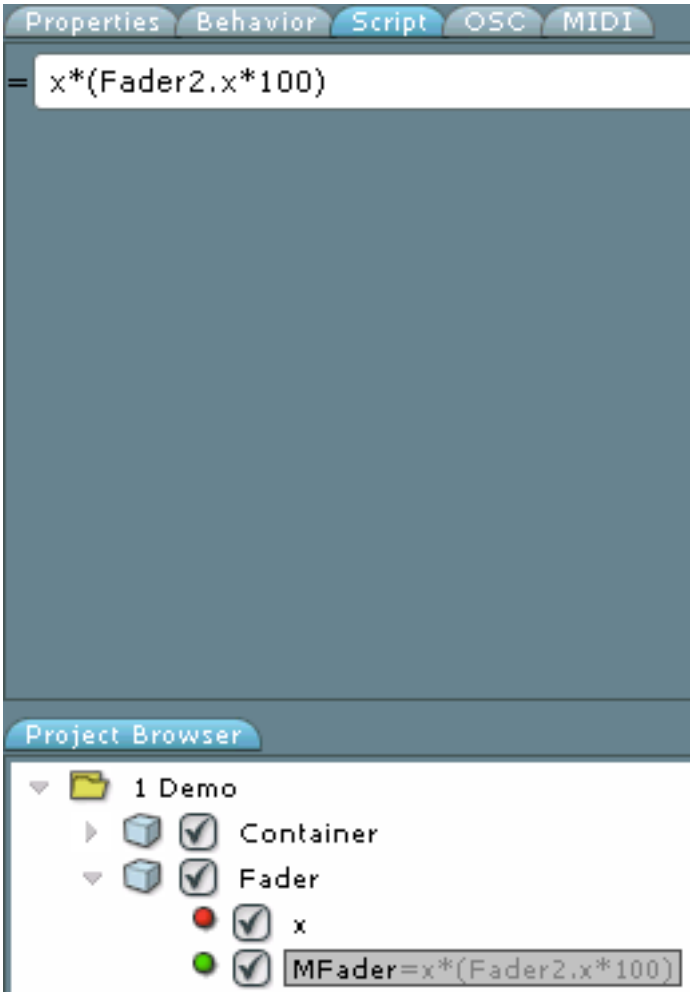
Now click on the Create Variable button below the Project Browser.



Type in the name **MFader** into the dialog and click **OK**. You should see the Variable named Mfader in the Project Browser, being **local** to the Fader.

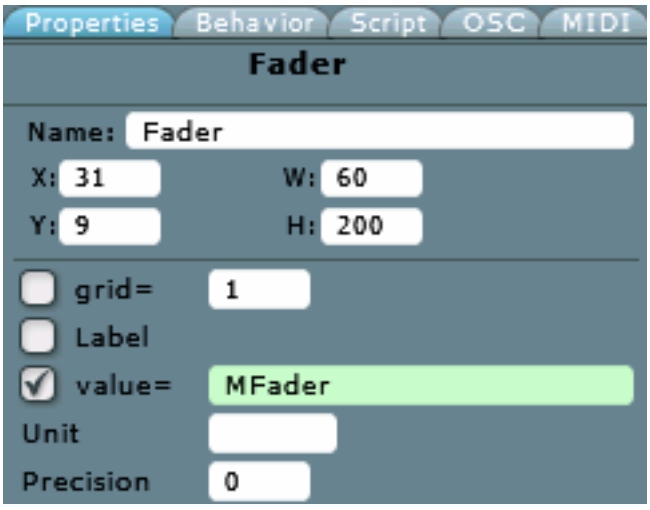


We want to scale the output of the first Fader with the value of the second. Do this by clicking on the fresh Variable in the Project Browser. Type **x*(Fader2.x*100)** into the Script field of the Variable. Make sure that the **checkbox** in front of the local Variable is **checked**. This ensures that the Variable is actually transmitted via OSC and MIDI. It should look like this:

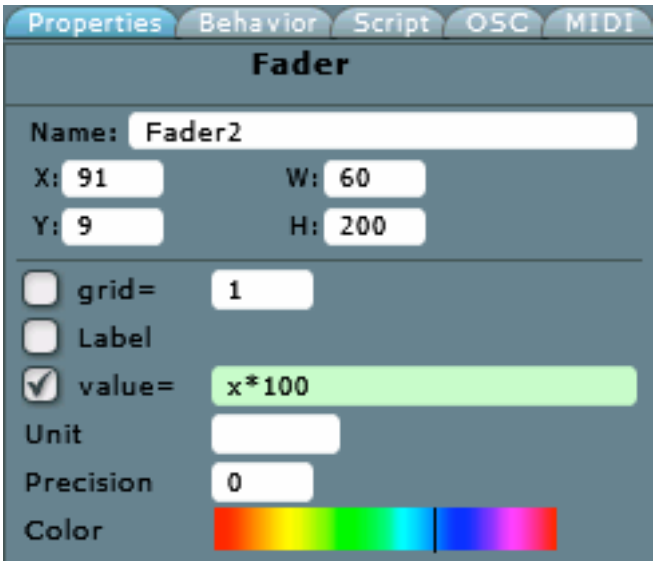


You might wonder why we addressed the Fader's value with x directly and not via Fader.x. This is possible because the Expression we created is **local** to the Fader so that it "knows" the address of its parent. If we would have created the Expression **globally**, we would have used the full address (Fader.x).

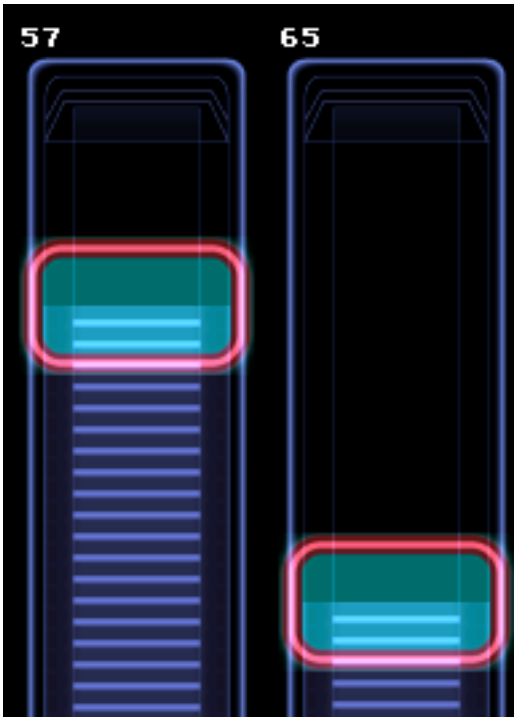
Of course you want to see the output of the Variable. Open the **General Properties** of the first Fader and check the **value flag**. Type **MFader** into the **value field**. As we don't care about decimal places this time, set the **Precision** to **0**.



For the **second Fader** also check the **value checkbox** and type **x*100** into the **value field**. The **Precision** can also be set to **0**.



If you now **move** the Faders you will see the relevant **values displayed** above. Move the second Fader and the value of the Variable will change, because it gets **multiplied** by a factor between 0 and 100 depending on the value of the second Fader (Fader2.x*100).



Please note, that the **displayed values** are not those actually produced by the Faders. They both still produce values **between 0 and 1**. Those values are combined in the calculation of the local Variable MFader that is displayed above the first Fader. The factor that's used for multiplication is displayed above Fader 2.

Now, what if you want to multiply with multiples of 10, only? The Fader has grid variable for quantization of the output.

We want to transmit the local Variable MFader via **MIDI**. This is easily done via the **MIDI panel** of the **first Fader**. The **Variables menu** contains the Variable **MFader** and we can choose it to be transmitted to MIDI Target 0 (or any other Target we want). Choose **Control Change** as **Message** and **Controller 10** as controller. Make sure that you **uncheck** the **Scale checkbox**, as we want the values to be transmitted as is and not extrapolated to values between 0 and 127.

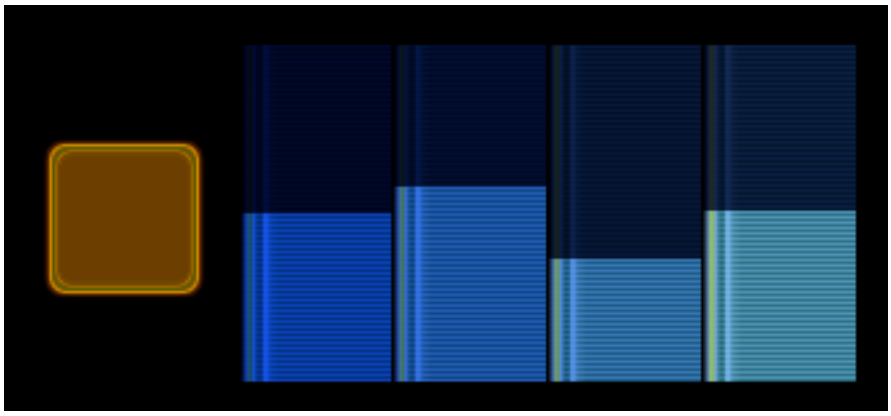
Now your Lemur will output values between 0 and 100 to MIDI Target 0 depending on the state of the two Faders.

3.5.4. Using Vector Variables

The Lemur doesn't only know data structures containing single values. There are also vector/list variables that consist of more than one value. **Vectors** are used by the MultiSlider, MultiBall, Pads, and Switches Objects.

This can be convenient when you need controls for ADSR envelopes or other tightly interrelated groups of parameters. Let's use a MultiSlider to produce values for an envelope working somewhere in a Synthesizer and trigger them with a Pads Object:

- Switch to the **1 Demo** Interface and click into the **Container**. There's some space left.
- Click the **Create** Object button and choose the **Pads**. Name the Object **Trig** and click OK.
- Click the **Create** Object button again, choose **MultiSlider** from the list of objects, name the object **Env**, and click OK.
- Click the **Behavior** panel of the MultiSlider, and set the number of sliders to **4**. Move and resize the MultiSlider so you can control each slider easily. Maybe something like this or a little bigger?



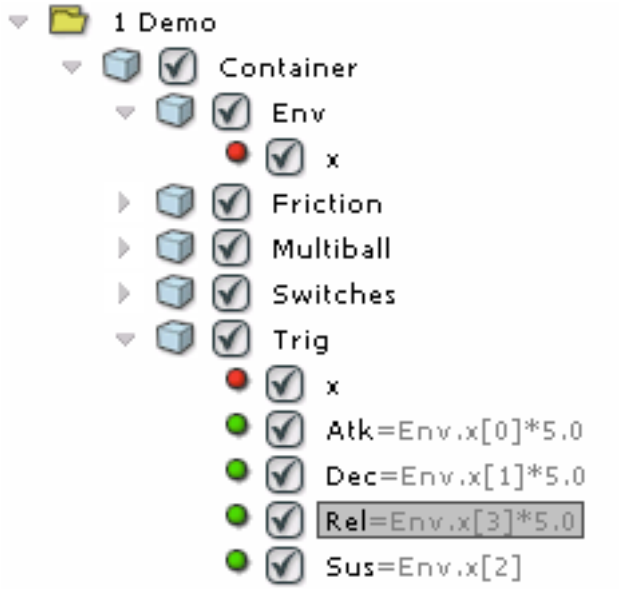
We're going to use the first slider for **Attack**, the second for **Decay**, the third for **Sustain**, and the fourth for **Release**. As with the x variables of all Lemur objects, the MultiSlider x variable ranges between 0 and 1. This is not really enough of a range for the time values of our envelope, so we need to scale these values. We'll also need to create three Expressions, one for each of the envelope's time values.

- Click on the **Trig** Object (Pads) in the Project Browser so that its name is selected.
- Click the **Create Expression** button to create a new, local Expression.
- Name the expression **Atk** and press return
- Click on the new expression in the Project Browser and change to its **Script panel**.
- In the script field enter the **Expression** shown below:

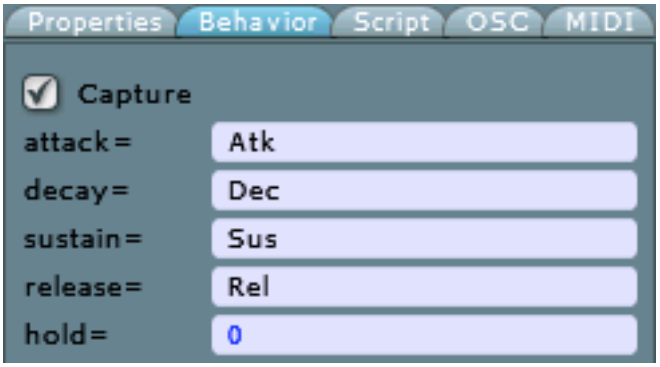
This means that the Atk Variable will use the value of the first slider (with the index of 0) of the **MultiSlider** we just created.

Create similar Expressions for **Dec** (decay), which will use **Env.x[1] * 5.0**, **Sus** (sustain), which will use **Env.x[2]**, and **Rel** (release), which will use **Env.x[3] * 5.0**. Note that we do not scale the value for Sustain, since this is just a value between 0 and 1.

When you're done, your Project Browser should look like this:



The final step is to enter these Variable names in the **Behavior** Properties of the Trig Object. Enter **Atk** for **Attack**, **Dec** for **Decay**, **Sus** for **Sustain**, and **Rel** for **Release** as shown below. As the Variables are local to the Trig Object, you don't have to write out the complete addresses (which would be Trig.Atk, Trig.Dec, etc.).



Now you can test your envelope by hitting the Pads Object. You don't even need an attached Synthesizer to see what's going on. Dial in the envelope values via the MultiSlider and you will see how the changes of the sliders affect the brightness envelope of the pad.

3.5.5. Using Custom MIDI Messages

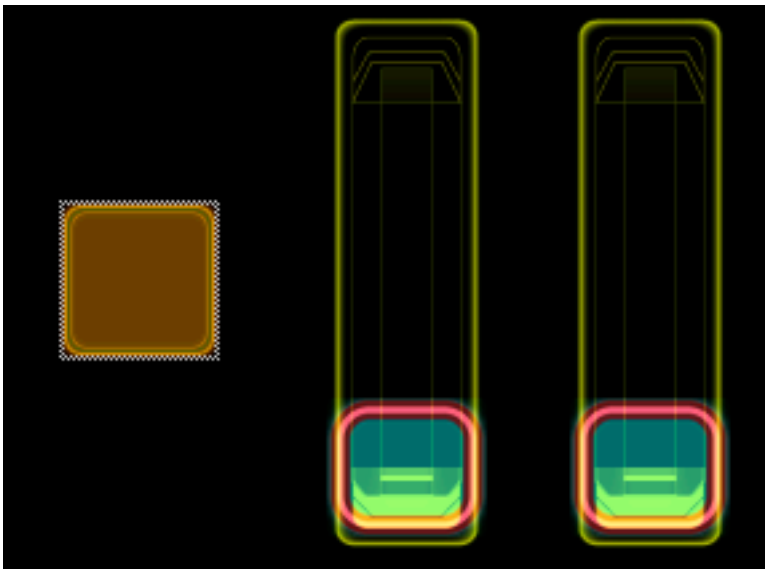
Custom MIDI messages provide simple means to generate very complex MIDI data by entering **Expressions** for all the **parameters** of the chosen **MIDI message**. With normal MIDI assignments you can only assign the selected variable to one parameter of the message. With Custom MIDI messages you gain complete freedom to assign any value or expression to any one of the different parameters of the message. And remember that expressions can use all values of all Objects in your Project.

For a Note On message the parameters would be:

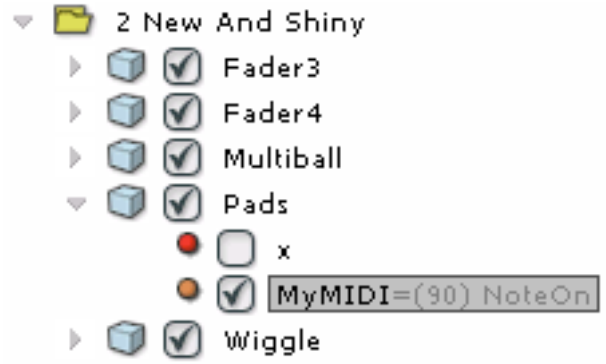
- **Pitch**
- **Velocity**
- **MIDI Channel**

You can even use an additional expression to control the triggerring of the MIDI message. There is also **Trigger Mode** menu for all the parameters. Please read chapter 3.4.4 for details.

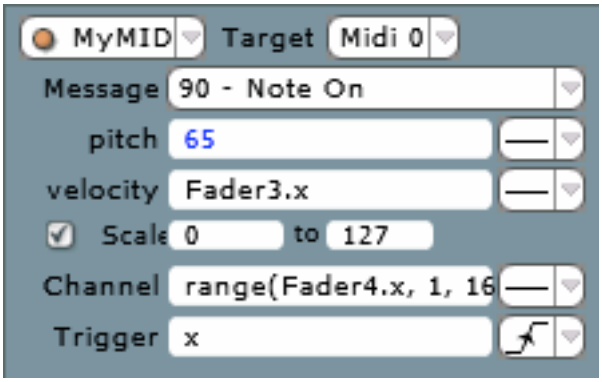
As a small example, try adding a **Pads** Object and **two Faders** (Fader3 and Fader4) to the second Interface of our Project.



Next, create a **Custom MIDI message** local to the Pads Object and call it **MyMIDI**.



Now look at the **MIDI panel** of the Custom Midi message and choose **Note On** from the message menu.



We want a **fixed pitch** for the generated note messages and set the pitch field to 65.

For **velocity** we use the value of **Fader3**, scaled to the MIDI range of **0** to **127** via the **Scale fields**.

The **trigger** field is set to x, which is the state of the Pad. By setting the trigger mode for this field to "up", the MIDI message is sent when x leaves 0, i.e when the pad is pressed.

Now we can trigger notes with the Pad and control the velocity with Fader3.

As a playful addition we assign the other Fader's value to the **MIDI channel.** Here we use **range function** to get values between 1 and 16.

Now you can choose the MIDI Channel for the outgoing note messages. This might be an exotic application but it's fun – stack up a pile of sounds in multimode in your favourite synth and trigger different sounds by moving the channel fader on your Lemur.

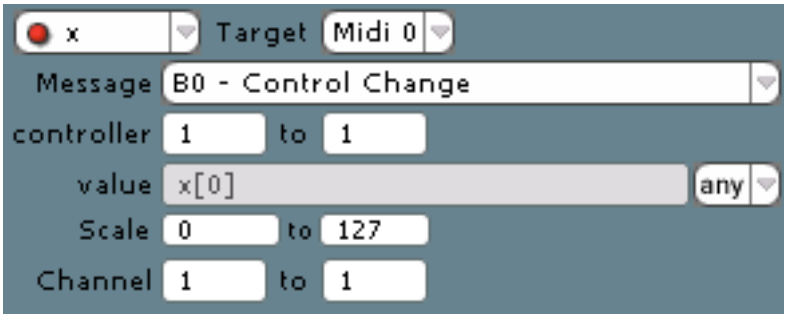
Have a look at the different **Trigger Modes** we use. Pitch, Celocity and Channel are set to **None**. Only a change in the Pads.x value will trigger transmission of the MIDI message.

Note that unlike usual variables, Custom MIDI messages cannot be controlled via OSC or MIDI. They only provide **outgoing data**.

3.5.6. Bi-directional Control

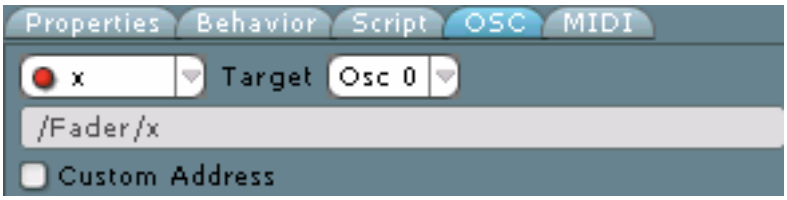
All Variables of a Lemur Project can also be easily **controlled by external software** or hardware via **MIDI** and **OSC**.

If you want to control a Variable via **MIDI**, just set up the necessary MIDI **input** for a **Target** in the MIDI settings. Then choose that MIDI Target for the Variable on the MIDI panel.



If you have, for example, a Fader's x value set up for transmission of MIDI controller 1 on MIDI Target 0, it will also be controllable via incoming controller 1 data on the input port chosen for that MIDI Target.

Similarly, you can control all values via **OSC**. The different values are controlled via their OSC addresses you already know.



For controlling a **Fader** via OSC, just set up the **OSC Target** for the **x value** on the OSC panel and produce the OSC messages on the Target side. In

the Fader's case you would send values between 0 and 1 to the OSC address /Fader/x to the current IP and port of the Lemur. If you've set up a custom OSC range for Fader.x, it will respond to values inside that range.

Please make sure that you think about the hierarchy of Objects on the Lemur. If the Fader is inside of a Container the address would be /Container/Fader/x.

Note that it's also possible to switch Interfaces via OSC. Just send an OSC message in the format "/interface InterfaceName" to the Lemur's IP (port 8000). In our case this would look like "/interface 1 Demo" for switching to the first one. For names that include spaces there is an additional consideration to be made. The OSC software has to send the name as a single string. This might involve putting it in quotes in the software. This is the case for Max/MSP.

For more about working with OSC, MIDI and bi-directional control please have a look at the extensive workshop chapters we offer on our website. There you'll learn everything about how the Lemur loves Max/MSP, Reaktor and other OSC enabled applications.

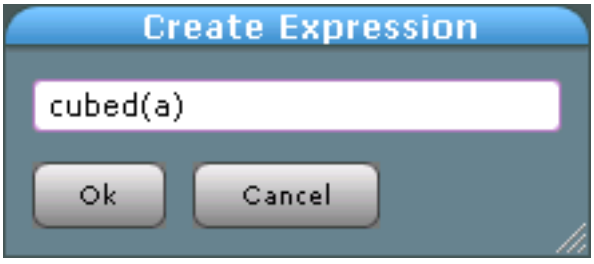
3.5.7. Defining and Using Functions

In addition to built-in functions and operators, you may define your own mathematical functions to use in other Lemur variables and expressions. Here's how to do it:

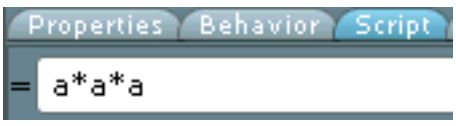
Click the Create Expression button to create a new Expression.

The Expression is created at the level of the hierarchy you have currently selected. If you have an Object selected it will be an Expression local to that Object. If you have the Global folder selected the Expression will go there.

- Type the name of the function and its arguments in parentheses. For example, to define a function named cubed that takes one argument, you would type:

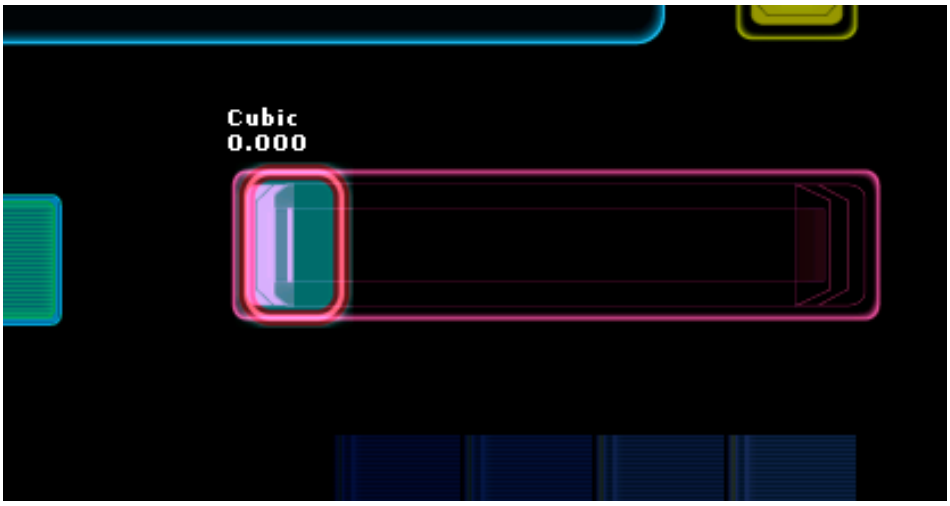


- Click OK and click on the Script panel of the new Expression.
- Type in the definition of your function as an expression. You can reference other variables if desired. The example below just multiplies the function argument a by itself twice to raise the input to the third power.

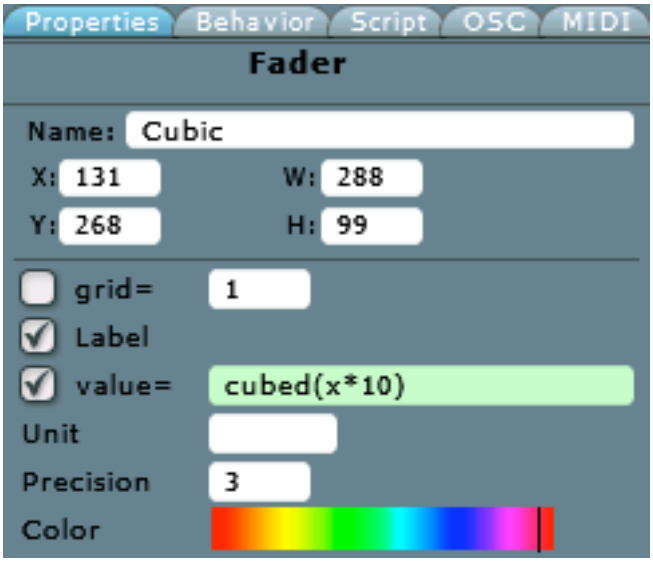


How to use the new function? We are running out of space with our Interfaces, but there's still an empty spot on the first one.

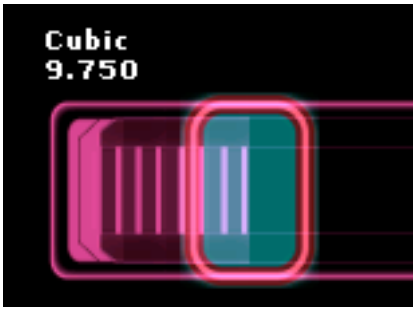
Switch to the **1 Demo** Interface and create a new **Fader** in the **Container**.



Make it a vertical one, change the colour to something different (in our case it's red). Also, **name** the Fader **Cubic** and check both the **label** and the **value** checkboxes.



Now switch to the Properties and set the value field to **cubed(x*10).** You are now using the Function we just created to raise the x value (which is multiplied by 10 beforehand) to the power of three. The variable argument of the Function is replaced by the Expression **x*10**.



Move the Fader and you will see that it displays an exponential range from 0 to 1000.

3.5.8. Vector Datacrunching

A **vector** (also known as an array or list), is a variable that holds more than one element (number). The Lemur expression syntax allows operations on vectors and individual access to their elements. You can access the data in certain Lemur Objects (MultiSlider, MultiBall, Pads, and Switches) as vectors. Typically each "**sub object**" is an element of a vector. For example, the x coordinates of the balls in a MultiBall object are accessed as

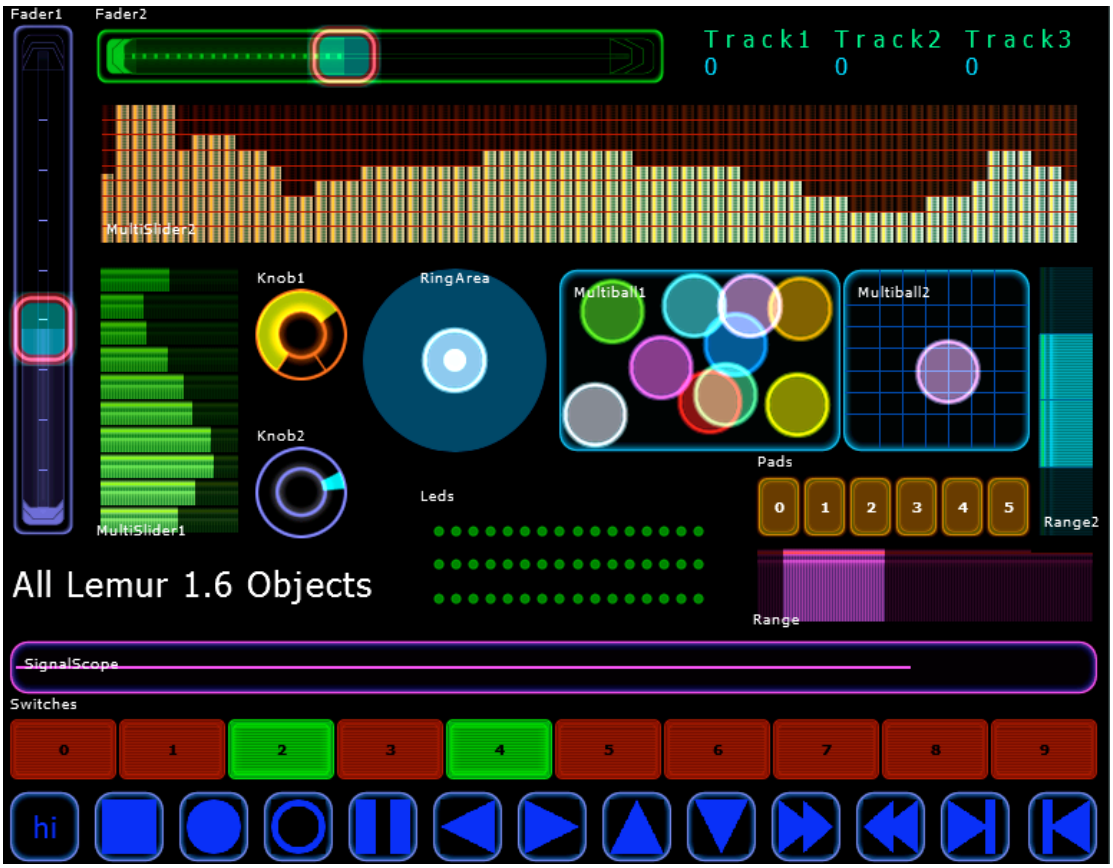
- myball.x[0]
- myball.x[1]
- etc.

Let's look at some of the operations you can perform on vectors. Assume we have a **MultiSlider** object called **moo** and a **MultiBall** object called **bar**. Below we have listed some expressions involving vectors and the results they would produce. Assume **moo.x contains {0.2 0.3 0.6}** and **bar.x contains {0.25 0.5}**

- moo.x * 3 Multiply all elements of moo.x by 3, returns {0.6 0.9 1.8}
- moo.x + 2 Add 2 to all elements of moo.x, returns {2.2 2.3 2.6}
- moo.x>0.5 Return a vector consisting of 1 or 0 depending on whether moo.x is greater than 0.5, returns {0. 0. 1.}
- moo.x + bar.x Add all elements of two vectors. The size of the result is the smaller of the two input vectors. Returns {0.45 0.8}
- moo.x[0] The first (index 0) element of moo.x, returns 0.2
- bar.x[1] The second element of bar.x, return 0.5
- moo.x[0.5] Interpolated value between bar.x[0] and bar.x[1], returns 0.3
- {moo.x, bar.x} Concatenates moo.x and bar.x, returns {0.2 0.3 0.6 0.25 0.5}
- {moo.x[0],bar.x[1.5]} Creates a new vector consisting of the first element of moo.x and the average of the second and third elements of bar.x, returns {0.2 0.625}

Please have a look at the **Parser Reference** for a complete overview of the possibilities.

Chapter 4 OBJECT REFERENCE



The Lemur contains **fourteen Objects** (and the utility Object Container) you can use to build interfaces. This section explains the features, options, and OSC message format of each Object. The Object names appear in the picker window you see when clicking the Object button at the bottom of the JazzEditor window.

Make sure to periodically check the JazzMutant website for updates, as the number of available Objects will grow in the future.

4.1.1. Guide to Object Reference Pages

Variables	Lists the built-in Variable names of the Object and what they represent. All built-in Variables range from 0-1 unless otherwise noted.
Dimensions	Lists the default and minimum dimensions (height and width) of the Object in pixels.
General Properties	Describes the properties shown in the General tab of the Properties inspector. Properties marked with a checkbox (☒) are on/off properties (typically check boxes). Otherwise, the properties can accept constants, Variables, or Expressions.

Behavior Properties — Documents the behavior properties of the Object—generally these are related to physical modelling features. Properties marked with a checkbox (☒) are on/off properties (typically checkboxes). Otherwise, the properties can accept constants, Variables, or Expressions.

4.1.2. Container

The Container is a utility Object helping to structure your Interfaces. You can enclose any number of Objects including other Containers. A Container can be transparent, meaning that anything lying "under" the contained Objects will show. A transparent Container also loses its frame.

You can also lock the Container, making it impossible to edit the Objects inside. The contained Objects also won't show up in the Project Browser anymore, unless you unlock the Container again.

Variables

none

Dimensions

Default	60 x 60 pixels
Minimum	32 x 32 pixels

General Properties

Name — The name of the Container. Must be used as a prefix for addressing the Objects inside of the Container (as in Container/Fader.x)

Lock — A locked Container prevents editing and display of its content in the Project Browser.

☒ Transparent — If checked, the Objects lying "under" the Container (but not belonging to its content) are shown and the Container's frame is not displayed anymore.

Color — Drag the color bar to change the frame color of the Container.

OSC Data

Lemur->Host — none

Host->Lemur — none

4.1.3. Custom Button

The Custom Button can act as a pad or switch. You can set the text of the button for on and off state. Another option is to have the state displayed by two different geometric forms to be chosen among .

Variables

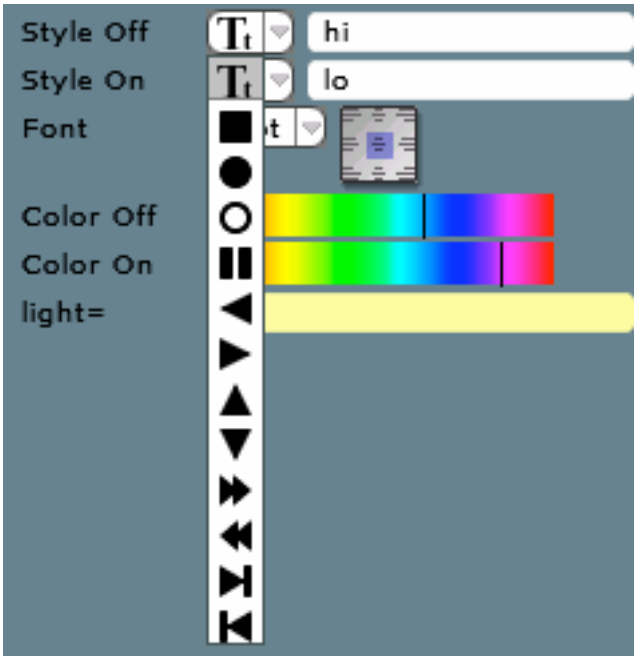
x A list of the envelope (brightness) values of the pads in the Object.

Dimensions

Default	60 x 60 pixels
Minimum	32 x 32 pixels

General Properties

Name — The name of the Custom Button Object that's also used as its address.



Style Off — A menu for choosing how the Off state is depicted. Choose between Text and 12 different symbols. If you choose Text, whitespace can be used in the text entry.

Style On — A menu for choosing how the On state is depicted.

Font — A menu for choosing the font size for the text. The font size ranges from 8pt to 24pt.

Alignment — A graphical menu for choosing the position of the text within the boundaries of the Object. You get a choice between 9 different positions. This feature does not work for the symbols.

Color Off/On — Dial in the color for the two respective button states.

Light — Can be a constant or any mathematical expression and controls the luminosity of your button. -2 means black, +2 means white, and you get to choose any decimal number in-between.

Behavior Properties

Mode — A menu for choosing if the Button works as a Switch or a Pad. A Switch changes state on touch and doesn't change back on removal of the finger. A Pad changes its state back if you remove your finger.

4.1.4. Fader

The Fader tracks your finger with a virtual cap and transmits one value corresponding to the position of the cap on the fader. The Fader can be oriented vertically and horizontally. Just grab a corner and drag it to change orientation.

Variables

x	The location of the cap. When the cap is at the top or right-most position of the fader (depending on orientation), the value is 1 by default. When it is at the bottom or at the left-most position, respectively, the value is 0.
Z	A flag variable for detecting if the Fader is being touched. 1 if touched, 0 if untouched. This is useful to emulate fader touch of control surfaces.

Dimensions

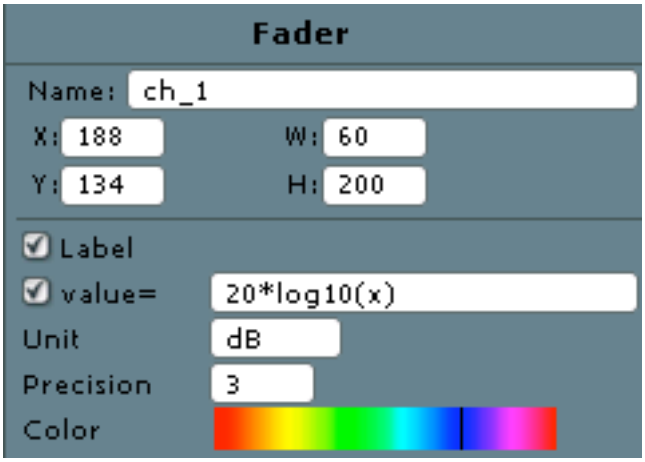
Default	60 pixels wide by 200 pixels high
Minimum	53 pixels wide by 200 pixels high

General Properties

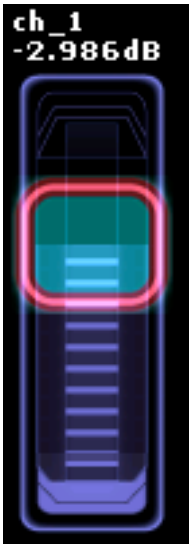
Name	The name of the Fader that's also used as its address.
☒ Label	If checked, the Object's name is displayed on the Interface.
☒ grid	If checked, the range of values produced by the Fader is quantized into [grid] steps. The maximum number of steps for the Fader is 33.
☒ Value	If checked, the current value of the Fader is displayed on the Interface. In addition, you can enter a formula for how the value is displayed. This does not affect the actual value sent by the Fader, which remains between 0 and 1.
Unit	This user-specified text is appended at the end of the value display. Use it to specify the type of value, as in dB or ms.
Precision	Specifies the number of decimal places for the value display. The default value is 3 and the maximum number is 6. This setting has no influence on the actual output of the Object. You have to scale the output using expressions or on the Target side.
Colour	Drag the colour bar to change the background colour of the fader. The cap always keeps a pink outline.

Properties Example

This collection of General properties can be used to configure a fader that displays values in decibels, where 0 dB is considered the maximum.



The resulting Fader Object looks like this:



Behavior Properties

☒ Capture — If Capture is checked, an Object will only react to cursors that were created inside its area. Even if the cursor later leaves the Object for another position, it will remain in control of the original Object, until it is destroyed eventually. When Capture is off, the old school way from previous versions is restored, meaning an Object will react to whatever cursor is present at any moment in its area.

Cursor Mode — If set to **Limited**, the Fader will respond to a new cursor only if the original one has been destroyed (i.e. finger is raised)

Get Newer means that whenever a new cursor appears inside the Object's area, it gains full control of the Object

In **Barycentric** mode each cursor, old and new, has the same amount of influence on the Object

In **Cap Only** mode, the Object acts like a conventional fader that doesn't react to cursors outside of the cap area

Physics — Can be one of three settings: None, Interpolate, or Mass-Spring.

If Physics is set to **None**, the Fader cap tracks one finger immediately. If other fingers touch the fader, they are ignored.

If Physics is set to **Interpolate**, the cap moves according to the value of Attraction from its current location to the location of your finger. Larger values for attraction (up to 1) cause the cap to move to the new finger location more quickly. When Attraction is set to 0, the cap cannot be moved by your finger.

If Physics is set to **Mass-Spring**, Attraction and Friction are both active. Friction ranges between 0 and 1. Lower values of friction mean that if the cap is moving it will tend to keep moving. With a value of 0, the cap will never stop moving. At a value of 1, the cap exactly follows your finger. Values of 1 for Attraction and Friction are essentially the same as if Physics is set to **None**.

Please consider that the Lemur will send the Fader's position constantly with lower values of Attraction and Friction in Mass-Spring mode.

Attraction — The amount of attraction the cursor (your finger) has on the Fader.

Friction — See Mass-Spring above.

4.1.5. Knob

The **Knob** Object emulates two types of knobs: Classic knobs which are constrained between a min and max value (0 and 1 by default), and endless encoders which can reach any value through multiple successive turns.

Variables

x — The current value of the knob, based on absolute angle position in Classic mode, or rotation count in Endless mode

General Properties

Name — The name of the Knob Object that's also used as its address.

☒ grid — If checked, the range of values produced by the Knob is quantized into [grid] steps. The maximum number of steps for the Knob is 33.

☒ Label — If checked, the Object's name is displayed above the Knob.

☒ Endless — If checked, the knob switches to endless mode.

☒Value — If checked, the current value of the Knob is displayed on the Interface. In addition, you can enter a formula for how the value is displayed. This does not affect the actual value sent by the Knob, which remains between 0 and 1.

Unit — This user-specified text is appended at the end of the value display. Use it to specify the type of value, as in dB or ms.

Precision — Specifies the number of decimal places for the value display. The default value is 3 and the maximum number is 6. This setting has no influence on the actual output of the Object. You have to scale the output using expressions or on the Target side.

Color — Drag the colour bar to change the colour of the Knob.

Behavior Properties

Cursor Mode — If set to **Limited**, the Knob will respond to a new cursor only if the original one has been destroyed (i.e. finger is raised)

Get Newer means that whenever a new cursor appears inside the Object's area, it gains full control of the Object

In **Barycentric** mode each cursor, old and new, has the same amount of influence on the Object

Control — In **Polar** mode, the Knob respond to the polar position of cursors, relative to its center point.

In **Linear** mode, the Knob responds to cursor drag on the vertical axis.

Physics — Can be one of three settings: None, Interpolate, or Mass-Spring.

If Physics is set to **None**, the Knob jumps instantly to the value dictated by the cursors' positions.

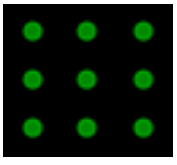
If Physics is set to **Interpolate**, the Knob rotates toward new values according to the value of Attraction. Larger values for attraction (up to 1) cause the Knob to reach a new value more quickly. As the Attraction value is lowered, the Knob takes longer to arrive at the finger position. When Attraction is set to 0, the Knob cannot be moved by your finger.

If Physics is set to **Mass-Spring**, Attraction and Friction are both active. Attraction works as described above under Interpolation. Friction ranges between 0 and 1. Lower values of friction mean that the Knob will tend to keep moving. With a value of 0, the Knob will essentially never stop moving. Values of 1 for Attraction and Friction are essentially the same as if Physics is set to Interpolate.

Attraction — See the discussion of Interpolation and Mass-Spring modes under Physics above.

Friction — See the discussion of Mass-Spring mode under Physics above.

4.1.6. Leds



The **Leds** Object is a two-dimensional array of Leds. It can be a single led or a whole matrix of them, all with their individual light and value parameters. Remember that those properties can be set to vectors when dealing with several columns or rows of Leds.

The Bar Graph mode makes a matrix or vector of Leds act as your typical bar graph: the Leds work together to graphically display the current value.

The Leds are transparent to touch but not transparent in terms of display. This means you can implement switch or pad functionality by placing a hidden Switches or Pads Object underneath the Leds.

Variables

value In Bargraph mode this value represents the percentage of the Leds being switched on. If Bargraph mode is off, only 1 and 0 are accepted as values. If you send a single value (0,1) it will switch all Leds to their on or off colors. If you have multiple rows/columns, you can use a vector to individually address the Leds.

General Properties

Name The name of the Leds Object that's also used as its address.

☒ Label If checked, the Object's name is displayed above all of the Leds.

Columns The number of columns of Leds contained in the Object. Only 16 columns of Leds can be set.

Rows The number of rows of Leds contained in the Object. Only 16 rows of Leds can be set.

Color Off Drag the colour bar to change the colour for the off-state of the Leds.

Color On Drag the colour bar to change the colour for the on-state of the Leds.

Light Can be a constant, a vector or any mathematical expression and controls the luminosity of your Objects. -2 means black, +2 means white, and you can choose any decimal number in-between.

Behavior Properties

none

4.1.7. Monitor

The Monitor produces no data when you touch it. Its purpose is to display information sent to the Lemur by your computer.

Variables

value Represents the value of the Monitor to be displayed. You can use a constant or any expression.

Dimensions

Minimum 50 pixels wide and 22 pixels high. The font size of the monitor is adjusted as the Object is resized.

General Properties

Name The name of the Monitor that's also used as its address.

☒ Label If checked, the Object's name is displayed above the value.

Value The checkbox is not functional. The text field next to Value represents the monitor's default value. Since any value can be sent to the monitor, there is no 0-1 limitation.

☒ Transparent	If checked, only the label and the value is displayed and the Monitor's background becomes transparent. This also makes it possible to use the Monitor as a text label anywhere on the Interface.
Unit	Appends arbitrary text to the end of the value display.
Precision	Number of floating-point digits that appear. A precision of 0 displays only the integer part of numbers the monitor receives.
Font	A menu for choosing the font size for the label and value of the Monitor. The font size ranges from 8pt to 24pt.
Alignment	A graphical menu for choosing the position of the text within the boundaries of the Object. You get a choice between 9 different positions.
Color	Drag the color bar to change the color of the label text. The numerical value display is always white.

Behavior Properties

None.

4.1.8. MultiBall

The MultiBall Object assigns each finger to track one of a number of balls (up to 10) in a rectangular space. Balls can either always be visible or only appear when you touch the space; the latter is called ephemeral mode. The brightness of the balls is sent as the z variable in the Object.

Variables

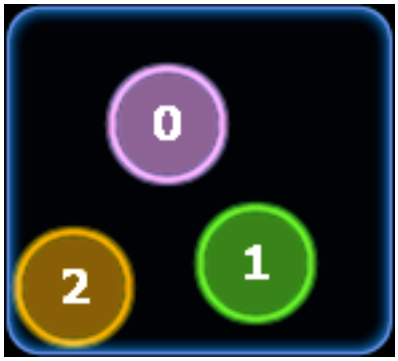
x	A list of the horizontal positions of all the balls.
y	A list of the vertical positions of all the balls.
z	A list of the brightness values of all the balls. Brightness values change only when the MultiBall Object is in ephemeral mode.

Dimensions

Default	128 x 128 pixels
Minimum	128 x 128 pixels

General Properties

Name	The name of the MultiBall Object that's also used as its address.
☒ Label	If checked, the Object's name is displayed above the balls.
Numbers	If checked a running number is displayed inside of each ball (see picture below)



Balls	Number of balls (1 to 10)
☒ grid	If checked, the range of values produced by the Balls is quantized into [grid] steps. The maximum number of steps for the MultiBall axis is 33.

Color — Drag the color bar to change the color of the outline around the MultiBall's rectangular space. The colors of the balls are fixed as follows:

1: faded purple
2: lime green
3: "lemur" yellow/orange (see screen shot above)
4: lemon yellow
5: red
6: teal blue
7: deep blue
8: sea green
9: intense purple
10: white/grey

Behavior Properties

☒ Ephemeral — If checked, the MultiBall behaves in a mode where the balls disappear until you touch it with one or more fingers. The brightness of the balls becomes the value of the z variable of the Object, and the way the brightness changes over time is controlled by an ADSR envelope (described below). When Ephemeral is not checked, the balls are always visible and their z values are constantly 1.

☒ Capture — If Capture is checked, an Object will only react to cursors that were created inside its area. Even if the cursor later leaves the Object for another position, it will remain in control of the original Object, until it is destroyed eventually. When Capture is off, the old school way from previous versions is restored, meaning an Object will react to whatever cursor is present at any moment in its area.

Cursor Mode — If set to **Limited**, the Balls will respond to a new cursor only if the original one has been destroyed (i.e. finger is raised)

In **Barycentric** mode each cursor, old and new, has the same amount of influence on the Object.

Get Oldest Ball: If all the balls are currently "possessed" by your fingers and a new cursor comes in, it will take control of the oldest ball (the one that lived the longest in Ephemeral mode or the first ball in normal mode).

Get Closest Ball: In the same situation, the new cursor gets attributed the ball that is the closest in distance.

Physics — Can be one of three settings: None, Interpolate, or Mass-Spring.

If Physics is set to **None**, balls move to finger positions immediately, and the settings of Attraction and Friction are ignored.

If Physics is set to **Interpolate**, balls move toward finger positions according to the value of Attraction. Larger values for attraction (up to 1) cause a ball to move to a finger position more quickly. As the Attraction value is lowered, the balls take longer to arrive at the finger position. When Attraction is set to 0, balls cannot be moved by your finger.

If Physics is set to **Mass-Spring**, Attraction and Friction are both active. Attraction works as described above under Interpolation. Friction ranges between 0 and 1. Lower values of friction mean that if a ball is moving it will tend to keep moving. With a value of 0, the balls will essentially never stop moving. At a value of 1, a ball will move only where you touch with your finger. Values of 1

for Attraction and Friction are essentially the same as if Physics is set to Interpolate.

Attraction — The amount of attraction the cursors (your fingers) have on the Balls.

Friction — See the discussion of Mass-Spring mode under Physic above.

Speed — This value multiplies the balls' speed after Physics computation by a user-defined expression. Input a singleton for the same effect on all balls, otherwise input a vector. Experiment with negative values for crazy effects.

About the MultiBall ADSR+H

The next four properties relate to the "brightness" or z variable envelope that can be used in ephemeral mode. You specify time values for the Attack, Decay, and Release in seconds as well as a Sustain level between 0 and 1. Unlike a keyboard ADSR, the MultiBall ADSR goes through its entire cycle when you touch the surface to trigger the appearance of a ball. If the ball is still tracking your finger when it reaches the end of the Decay portion of the envelope, the z value (brightness) will remain at the sustain level until you lift your finger off the surface. At that point (or before, if your finger has already left the surface), the Release portion of the envelope will be triggered, and the z variable will return to 0.

Attack — Applies only when using ephemeral mode. The Attack value specifies the number of seconds over which the z variable (brightness of a ball) increases from its initial value of 0 to a maximum of 1 after you touch the screen. As an example, if the Attack value is 0, the ball will be at full brightness the moment you touch the screen.

Decay — Applies only when using ephemeral mode. The Decay value specifies the number of seconds over which the brightness will decrease after the initial Attack portion of the envelope has completed. During the Decay portion of the envelope, the z variable (brightness of a ball) will decrease from 1 to the level set by the Sustain value.

Sustain — Applies only when using ephemeral mode. The sustain value is the level (between 0 and 1) at which the brightness of the ball will remain as long the MultiBall Object is tracking your finger within its space. The Sustain level is reached after the Attack and Decay portion of the envelope have completed. If your finger lifts up from the touch surface before the completion of the Attack and/or Decay portion of the envelope, the Release portion of the envelope is triggered immediately after the Decay portion completes, and the brightness ultimately goes to 0.

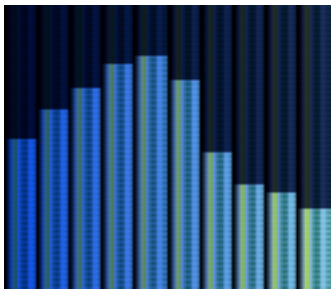
Release — Applies only when using ephemeral mode. The Release value specifies the number of seconds over which the brightness of a ball will decrease from its Sustain level to 0, starting at the moment that you lift up ("release") your finger from the touch surface.

Hold — Its effect is similar to a sustain pedal, freezing the Object's state as long as its value is 1. When set to 0, if has no effect. This means this parameter should be used with a mathematical expression depending on other Objects. For instance, if you have a Switch Object named Sustain in your interface, you can set the hold parameter of a Pad to Sustain.x so the Switch gets the ability to freeze the current lightness.

HoldX — If 0, this has no effect. Any value greater than 0 freezes the respective ball on its current position on the x-axis. The y-axis remains active. Use a vector, if you want to affect specific balls.

HoldY — If 0, this has no effect. Any value greater than 0 freezes the respective ball on its current position on the y-axis. The x-axis remains active. Use a vector, if you want to affect specific balls.

4.1.9. MultiSlider



The MultiSlider Object tracks movement across an array of sliders. You can "wipe" all the faders to a set value with one horizontal gesture. This is pretty hard to do with real—or virtual—faders.

Variables

x — A list of the vertical positions of all the individual sliders.

Dimensions

Default 100 x 100 pixels

Minimum 64 pixels wide x 40 pixels high

General Properties

Name — The name of the MultiSlider Object that's also used as its address.

☒ grid — If checked, the range of values produced by the Sliders is quantized into [grid] steps. The maximum number of steps for the MultiSlider is 33.

☒ Horizontal — Swaps the orientation of the MultiSlider from vertical to horizontal.

☒ Label — If checked, the Object's name is displayed above the sliders in the lower left-hand corner.

Slider — Number of sliders (1 to 64)

Color — Drag the color bar to change the "thematic" color of the sliders. The foreground uses a gradient based on this color, and the background is a darker version of the color.

Light — Can be a constant, a vector or any mathematical expression and controls the luminosity of your Objects. -2 means black, +2 means white, and you get to choose any decimal number in-between.

Behavior Properties

☒ Capture — If Capture is checked, an Object will only react to cursors that were created inside its area. Even if the cursor later leaves the Object for another position, it

will remain in control of the original Object, until it is destroyed eventually. When Capture is off, the old school way from previous versions is restored, meaning an Object will react to whatever cursor is present at any moment in its area.

☒ Physics — If checked, the MultiSlider emulates the physics of an Object similar to a plucked string anchored at the left and right sides of the array of sliders. Your fingers "pluck" the string by lifting it up in one or more places. The values of the sliders ramp up to meet your fingers and track them as they move. Lifting your finger(s) from the surface releases the string, and its subsequent behavior is determined by the Tension, Friction, and Height values.

Tension — A value between 0 and 1 corresponding to the tension on a string. As tension increases, the frequency of oscillation of the string increases. Increasing the tension is something like turning the tuning peg of a guitar to raise the pitch of a string.

Friction — A value between 0 and 1 corresponding to the damping on a string. As friction increases, the damping on the oscillation increases. With large friction values, the string returns to its resting position quickly. With smaller friction values, the string may oscillate for a long time.

Height — When Gravity mode is enabled, the height (0 to 1) is the value of the initial and resting position of the string. When Gravity mode is not enabled, the height is the initial value of the MultiSlider after it is initialized.

4.1.10. Pads

The Pads Object is a two-dimensional array of buttons that are triggered by touch. They are intended to trigger events instead of represent state, since they eventually return to an "off" value after you touch them.

Variables

x — A list of the envelope (brightness) values of the pads in the Object.

Dimensions

Default 100 x 100 pixels

Minimum 32 x 32 pixels times the number of rows and columns

General Properties

Name — The name of the Pads Object that's also used as its address.

☒ Label — If checked, the Object's name is displayed above the pads, which are "squished" to accommodate the text.

☒ Numbers — If checked, each pad is labelled with a number corresponding to the order in which its value is transmitted, starting with 0 as shown below.

Columns — The number of columns of pads contained in the Object. There does not appear to be a maximum number of columns, but only 25 columns of pads can be seen on the screen at once. The minimum width of a pad is 32 pixels, and the Object will be resized if necessary to maintain the minimum pad height.

Rows — The number of rows of pads contained in the Object. There does not appear to be a maximum number of rows, but only 19 rows of pads can be seen on the screen at once. The minimum height of a pad is 32 pixels, and the Object will be resized if necessary to maintain the minimum pad height.

Color Off — Drag the colour bar to change the colour for the off-state of the Pads.

Color On — Drag the colour bar to change the colour for the on-state of the Pads.

Light — Can be a constant, a vector or any mathematical expression and controls the luminosity of your Objects. -2 means black, +2 means white, and you get to choose any decimal number in-between.

Behavior Properties

☒ Capture — If Capture is checked, an Object will only react to cursors that were created inside its area. Even if the cursor later leaves the Object for another position, it will remain in control of the original Object, until it is destroyed eventually. When Capture is off, the old school way from previous versions is restored, meaning an Object will react to whatever cursor is present at any moment in its area.

About the Pads ADSR+H

The four behavior properties of a Pads Object relate to its "brightness" or x variable envelope. You specify time values for the Attack, Decay, and Release in seconds as well as a Sustain level between 0 and 1. Unlike a keyboard ADSR, the Pads ADSR goes through its entire cycle when you touch the surface to trigger it. If your finger is still down on a pad when the envelope reaches the end of its Decay portion, the x value (brightness) will remain at the sustain level until you lift your finger off the surface. At the release point (or before, if your finger has already left the surface), the Release portion of the envelope will be triggered, and the x variable will return to 0.

To implement an immediate response trigger, set the Attack, Decay, and Release to 0, and the Sustain to 1.

Attack — The Attack value specifies the number of seconds over which the x variable (pad brightness) increases from its initial value of 0 to a maximum of 1 after you touch the screen. As an example, if the Attack value is 0, the pad will be at full brightness the moment you touch the screen. An attack value of 10 means the pad will take 10 seconds to reach the full value.

Decay — The Decay value specifies the number of seconds over which the x variable (pad brightness) will decrease after the initial Attack portion of the envelope has completed. During the Decay portion of the envelope, the x variable (pad brightness) will decrease from 1 to the level set by the Sustain value.

Sustain — The Sustain value is the level (between 0 and 1) at which the x variable (pad brightness) will remain as long your finger is touching the pad. The Sustain level is reached after the Attack and Decay portion of the envelope have completed. If your finger lifts up from the touch surface before the completion of the Attack and/or Decay portion of the envelope, the Release portion of the envelope is triggered immediately after the Decay portion completes, and the brightness ultimately goes to 0.

Release — The Release value specifies the number of seconds over which the x variable (pad brightness) will decrease from its Sustain level to 0, starting at the moment that you lift up ("release") your finger from the touch surface.

Hold — Its effect is similar to a sustain pedal, freezing the Object's state as long as its value is 1. When set to 0, if has no effect. This means this parameter should be used with a mathematical expression depending on other Objects. For instance, if you have a Switch Object named Sustain in your interface, you can set the hold parameter of a Pad to Sustain.x so the Switch gets the ability to freeze the current lightness.

4.1.11. Range

The Range Object is a slider with adjustable length. Touch either end to change the width/height of the range.

Variables

x — A list of the vertical positions of all the individual sliders.

Dimensions

Default — 60 x 120 pixels

Minimum — 20 pixels wide x 120 pixels high (vertical mode)

General Properties

Name — The name of the Range Object that's also used as its address.

☒ grid — If checked, the range of values produced by the Sliders is quantized into [grid] steps. The maximum number of steps for the Range Object is 33.

☒ Horizontal — Swaps the orientation of the Range Object from vertical to horizontal.

☒ Label — If checked, the Object's name is displayed within the Objects boundaries.

Color — Drag the color bar to change the "thematic" color of the sliders. The foreground uses a gradient based on this color, and the background is a darker version of the color.

Light	Can be a constant, a vector or any mathematical expression and controls the luminosity of your Objects. -2 means black, +2 means white, and you get to choose any decimal number in-between.

Behavior Properties

☒ Capture	If Capture is checked, an Object will only react to cursors that were created inside its area. Even if the cursor later leaves the Object for another position, it will remain in control of the original Object, until it is destroyed eventually. When Capture is off, the old school way from previous versions is restored, meaning an Object will react to whatever cursor is present at any moment in its area.
☒ Physics	If checked, the Range boundaries snap back to user-defined positions (that can evolve based on other variables) when you release them. Physics behavior is determined by the Tension, Friction, and Min_Height and Max_Height values.
Tension	A value between 0 and 1 corresponding to the tension of the string that links the boundaries to their snap-back positions. You can enter a vector to specify different tensions for the two extremities.
Friction	A value between 0 and 1 corresponding to the damping on a string. As friction increases, the damping on the oscillation increases. With large friction values, the string returns to its resting position quickly. With smaller friction values, the string may oscillate for a long time. You can enter a vector to specify different frictions for the two extremities.
Min_Height	When Physics mode is enabled, the height (0 to 1) is the value of the initial and resting position of the mininum boundary of the Range.
Max_Height	When Physics mode is enabled, the height (0 to 1) is the value of the initial and resting position of the maximum boundary of the Range.
drag	If drag is not zero, the current range can be dragged with the finger without changing its limits. You may use this with a switch to toggle the behaviour of your Range Object (put Switch.x into the drag field). The field can contain any expression.

4.1.12. RingArea

The RingArea tracks your finger inside a circular space. It reports the X and Y coordinates of a ball that can be programmed to have a variable degree of attraction toward a central point. You can specify the location of the attraction point within the circular space.

Variables

x The horizontal position of the ball.

y The vertical position of the ball.

Dimensions

Default 128 x 128 pixels

Minimum 128 x 128 pixels

General Properties

Name The name of the RingArea Object that's also used as its address.

☒ Label If checked, the Object's name is displayed above the circular space.

Color Drag the color bar to change the background color of the circular space. The ball and attraction point are lighter variants of this color.

Behavior Properties

☒ Capture If Capture is checked, an Object will only react to cursors that were created inside its area. Even if the cursor later leaves the Object for another position, it will remain in control of the original Object, until it is destroyed eventually. When Capture is off, the old school way from previous versions is restored, meaning an Object will react to whatever cursor is present at any moment in its area.

Attraction A value from 0-1 representing the speed of the ball from its attraction point to your finger when you touch the Object, and from your finger to the attraction point when you release your finger.

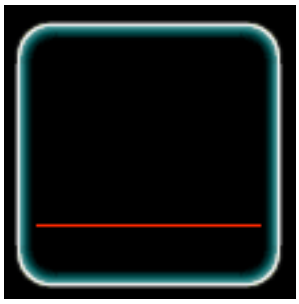
Friction A value from 0-1 representing the stickiness of the movement to either your finger or the attraction point. Lower values of friction make the ball overshoot the attraction point when it approaches, causing bouncing and/or oscillation.

Attractor X A value from 0-1 representing the horizontal location of the attraction point within the Object's rectangle. A zero value of Attractor X is at the left edge and a value of 1 is at the right edge.

Attractor Y A value from 0-1 representing the vertical location of the attraction point within the Object's rectangle. A zero value of Attractor Y is at the bottom edge and a value of 1 is at the top edge.

The exact position values transmitted by the Lemur when the ball arrives at the attraction point is partially dependent on the size of the RingArea. The ball never leaves the circular area, but as the Object grows larger in size, the attraction points can be located outside the circular area when Attractor X and Attractor Y are given very large or very small values.

4.1.13. SignalScope



The SignalScope displays values of other Objects and variables on your Lemur. The "trace" shows a recent history of the value of what you are monitoring.

The SignalScope is transparent to touch, meaning that you can place it on top of Objects and still interact with them.

Variables

None

Dimensions

Default 100 x 100 pixels

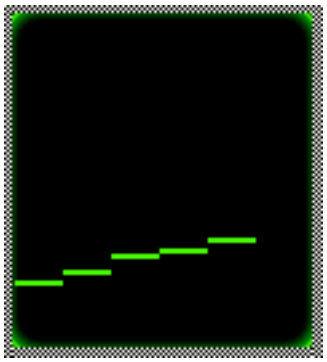
Minimum 32 x 32 pixels

General Properties

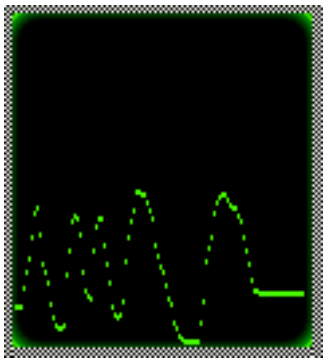
Name	The name of the SignalScope Object that's also used as its address.
☒ Label	If checked, the Object's name is displayed in the scope area.
☒ Mode XY	If checked, the signal scope shows both an X and Y value plotted against each other. If unchecked, the X value is time, shifting Y values to the left.
Color	Drag the bar to change the color of the scope and its frame.

Behavior Properties

X =	If Mode XY is enabled, X can be the value of a variable or a constant, otherwise it is assigned to time against which the Y value will be plotted.
Y =	A variable or constant expression that will be periodically evaluated and plotted against X. For example, to plot the x variable of a Fader Object called chan1, enter chan1.x in the text field.
Time base =	Corresponds to the time (in seconds) displayed on the scope. As the Time base increases, individual elements of the graph will decrease in width as more of the "past" is shown. The picture below shows a SignalScope displaying a tenth of a second of history. It is set to time base of 0.1:



And here is a SignalScope showing 2 seconds of history. Its time base is set to 2.0.



4.1.14. SurfaceLCD

The SurfaceLCD Object emulates traditional control surfaces' LCD, by making use of the MIDI input capabilities of the Lemur. With his Object you can display track names, parameter changes, timecode, all updated in real-time by your Digital Audio Workstation (DAW). The SurfaceLCD Object is compatible with the Mackie Control surface protocol.

Variables

None This Object has no value to be mapped to MIDI and OSC. It displays information coming from a DAW.

General Properties

Name The name of the SurfaceLCD Object that's also used as its address

☒ Transparent If checked, the background of the SurfaceLCD disappears to obtain a see-through appearance. If unchecked, the background color is set by the Top color slider.

Target This lets you select the MIDI Target number for information display. First, setup a MIDI target with an input of your liking inside the JazzEditor settings. Then, configure your DAW to send Mackie Control surface data to this MIDI input. Refer to the bundled examples for mappings of control Objects compatible with control surfaces protocols known by your DAW.

Display This lets you select one of three LCD modes common to conventional control surfaces.

In Main LCD mode, the Object displays Track information.

In **Timecode** mode, the Object displays up-to-date Timecode information.

In **Assignment** mode, the Object displays a three character code describing the current assignment of other mapped control Objects.

Refer to your DAW's manual for more information on specific display functions.

Behavior Properties

None

4.1.15. Switches

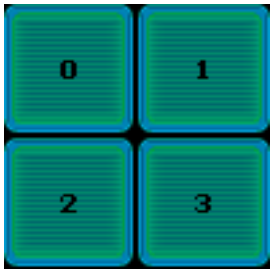
The **Switches** Object is a two-dimensional array of toggle switches. Their values reverse when you touch them. Switches can be used to represent and transmit one or more on-off states.

Variables

x	A list of the on-off values of the switches in the Object. The list starts with the top-left corner, and traverses the first row before starting at the beginning of the second row.

General Properties

Name	The name of the Switches Object that's also used as its address
☒ Label	If checked, the Object's name is displayed above all of the switches.
☒ Numbers	If checked, each switch is labelled with a number corresponding to the order in which its value is transmitted, starting with 0 as shown below.



☒ Radio	In Radio mode, only one switch can be turned on at any particular time. Turning on any switch turns all the others off.
Columns	The number of columns of switches contained in the Object. There does not appear to be a maximum number of columns, but only 16 columns of pads can be seen on the screen at once. The minimum width of a pad is 32 pixels, and the Object will be resized if necessary to maintain the minimum pad height.
Rows	The number of rows of switches contained in the Object. There does not appear to be a maximum number of rows, but only 16 rows of pads can be seen on the screen at once. The minimum height of a pad is 32 pixels, and the Object will be resized if necessary to maintain the minimum pad height.
Color Off	Drag the colour bar to change the colour for the off-state of the Switches.

Color On — Drag the colour bar to change the colour for the on-state of the Switches.

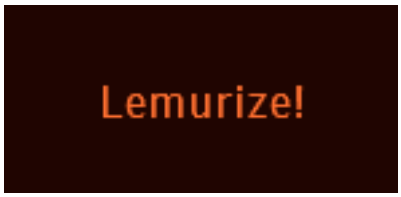
Light — Can be a constant, a vector or any mathematical expression and controls the luminosity of your Objects. -2 means black, +2 means white, and you get to choose any decimal number in-between.

Behavior Properties

☒ Capture — If Capture is checked, an Object will only react to cursors that were created inside its area. Even if the cursor later leaves the Object for another position, it will remain in control of the original Object, until it is destroyed eventually. When Capture is off, the old school way from previous versions is restored, meaning an Object will react to whatever cursor is present at any moment in its area.

Paint — If this flag is active, you can "paint" on an array of switches by dragging your finger around. If paint is inactive, a touch only toggles the switch you hit first and dragging the finger around has no further effect.

4.1.16. Text



The Text Object produces no data when you touch it. Its purpose is to display arbitrary text including whitespace characters typed in via the JazzEditor.

Variables

None — This Object has no value to be mapped to MIDI and OSC. It displays information defined by the user.

Dimensions

Minimum — 50 pixels wide and 22 pixels high. The font size of the text can be chosen (see below).

General Properties

Name — The name of the Text Object that's also used as its address.

☒ Transparent — If checked, only the text is displayed and the Object's background becomes transparent.

Text — Type in any text you want the Object to display for you.

Font — A menu for choosing the font size for the displayed text. The font size ranges from 8pt to 24pt.

Alignment — A graphical menu for choosing the position of the text within the boundaries of the Object. You get a choice between 9 different positions.

Color — Drag the color bar to change the color of the text and its background (if the Transparent flag is not set).

Light — Can be a constant or any mathematical expression and controls the luminosity of your text. -2 means the text is invisible, +2 means white, and you get to choose any decimal number in-between.

Behavior Properties

Object Reference

None.

Chapter 5 PARSER REFERENCE

The parser is the part of the Lemur's brain that crunches away on the Math you throw at it. It is used to do simple stuff like scaling an internal value or adding some Variables. But there's more. Due to a set of internal Operators and Functions and the possibility to create your own, custom Functions, you can construct very complex Projects. They may include a lot of internal calculation and interdependences between Objects helping in building the perfect, interactive Interface.

5.1.1. Value Colouring

There is a colour code for showing the type of value in a variable or expression. The colour changes automatically when you start typing into the respective fields:

- Black means a standard expression that is evaluates 60 times a second (every 16ms).
- Red indicated an invalid expression.
- Blue is a constant, like 10+2.

This is useful to check whether an expression is well formed. If it goes black, you know it can be calculated – which of course still doesn't mean it does what you want it to...

5.1.2. Vectors, Singletons and Operators

The Parser handles two kinds of variables: vectors and singletons.

a vector : {0, 2, 0.5}

a singleton : 3

If you want to access the items of a vector separately, use brackets [].

a = {1,3,4}

a[0] = 1

a[2] = 4

This also works for multiple items:

a[0,2] = {1, 4}

At the heart of the Lemur's parser lie the Operators and Functions. They are used to construct expressions that make the Lemur such a versatile animal.

The operators are basic mathematical critters that you should know from your Maths classes (you do remember your Maths classes, do you?)

+, -, *, /, ^

% is the modulo operator used to calculate the remainder of a division.

10%3 = 1

This can be very useful to derive "clean" values from the time variable (see below for a definition of Lemur time):

time%1 = 0.25 (if time = 2043.25)

5.1.3. Functions

Functions are the other founding stone of expressions in the Lemur. There is a variety of internal (pre-defined) Functions but you can also define your own Function.

You can create a Function by setting up a variable named xxx(a,b) and set its value as something like a-b+2.

The Functions can then be taken as the value for other Variables, where they receive "real" Lemur values as arguments. You could use the above function by making a Variable named yyy with the value xxx(Fader.x,Multiball.y). In this case a and b are substitutes by the current values of Fader.x and MultiBall.y.

Please note that most Functions that can act on singletons can adapt to vectors as well. For instance :

round(1.2) = 1

round({1.2, 3.8, 0.1}) = {1, 4, 0}

5.1.4. Expressions

Using all those bricks, you can create named Expressions (Variables) or use Expressions as Objects' parameters (attraction, etc.)

When entering a vector as a parameter for a multi-state Object (Switches, MultiBall, Leds, Pads), the parser will do its best to map each item in the vector to each sub-object

For instance :

entering {0.1, 0.5, 1.0} as the attraction for a Multiball that has 3 balls sets :

attraction for Ball 0 : 0.1

attraction for Ball 1 : 0.5

attraction for Ball 2 : 1.0

When mapping a vector Variable to MIDI (the x parameter of a Multiball with 10 balls, or a user-created Expression that evaluates as a vector), the parser will try to stretch the mapping to all the items.

For example:

A custom Expression called "vec" with value set to :

{Ball.x+Ball.y, Ball.x-Ball.y}.

If you now go into the MIDI panel and want to map the Expression to a MIDI controller, you just have to put controller 10 into the first controller field and the second field will automatically be set to 11.

5.1.5. Accessing Variables

An Expression that is local to an Object can access the x parameter of the Object just by its name "x":

scaled_x = x

If the Expression is created outside of the Object (local to another Object, or in the global folder, or at root level of a Container) the parent of the x Variable must be included in the address:

scaled_x = Fader.x

WARNING : Containers are taken into account in the hierarchy. If the Fader is inside container "Cont", all Expressions outside of the container must access it with "Cont.Fader.x"

Expressions inside of the same container can omit the first part and access it directly via "Fader.x"

5.1.6. Lemur internal clock

The Lemur works with a clock speed of **60 ticks per second**. This means that every **16 ms** all states of Objects, Expressions and Functions are evaluated. These 16 ms constitute one Lemur **frame**.

There are some special considerations, though, as the Lemur's brain tries to avoid unnecessary work. When it comes to deciding whether the value of a Variable is going to be transmitted via MIDI or OSC, the Lemur first checks if the Variable has **changed since the last frame**. If it hasn't changed, it's not going to be transmitted.

This has implications if you want to **transmit constant controller values** via MIDI (or OSC), e.g. via a Pads Object. It does not work to just put in a constant for your controller value, as the Lemur will refuse to send it after the first activation of the Pads Object. The value didn't change so it is not worth sending it.

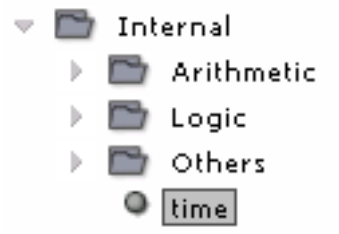
Let's assume you want to transmit the value **64** every time you hit the Pad.

Simply create a **Variable** (we called it "changer") local to the Pads Object and fill in **64*x** in the Variable's script field. Now switch to the MIDI panel, choose the new Variable and switch the **Trigger Mode** menu to **+.** For **Scale** you fill in **0** and **64**. That's it.

When the Pad is on, the Variable will have the value 64, when the Pad is off it will be 0 - it always changes. The Trigger Mode only allows transmission for a change from 0 to positive, so only the value 64 will get transmitted via MIDI.

5.1.7. Lemur Time

If you look at the **Internal** folder in the Project Browser, you will see a pre-defined Variable called **time**.



The time Variable is a millisecond value you can use for creating time-varying behaviour in your Lemur. It represents the **number of milliseconds** since your Lemur was turned on and it resets to 0 every hour. The time value is intended to be manipulated by mathematical Operators, particularly by multiplication (*), division (/), and by the modulo operator (%).

By multiplying time by a value greater than 1, you produce a sequence of values that increase faster than clock time. By multiplying time by a value less than 1, you will produce a sequence of values that increase more slowly than clock time. You can use the modulo Operator (%) to create a repeating sequence of values that resembles a sawtooth wave. For example time % 1 produces a ramp from 0 to 999 that occurs over the course of a second.

Here are some example Expressions created using the time variable:

5.1.7.1. Time Code Example

Use Monitor Objects to display the values of the following expressions:

Frames = (time % 0.25) * 100

Hours = floor(time/3600)

Minutes = floor(time/60) % 60

Seconds = floor(time % 60)

Create global Variables for each of these Expressions. Then you can use the Variables in other Lemur Objects. For example, here is the Properties Browser view for a monitor Object that would display a global Seconds variable.

Even if you are not connected to a Lemur, the resulting Monitor Object will begin changing immediately after you enter a time-based variable.

5.1.7.2. LFO Example

This is an Expression using the time Variable that generates an LFO.

LFO = sin(time * 2) *0.5 + 0.5

You can display this in a SignalScope.

You can assign time-based variables to properties in Lemur Objects (such as Friction or the ADSR envelopes) to create Objects that change over time. However, you can't assign them to the values of Objects (other than the Monitor and SignalScope), so you can't use the time variable to "animate" the purple ball in an Area.

5.1.8. Operators and Functions Reference

The following is a list of all internal Functions and Operators the Lemur parser understands.

Try them out by creating a monitor Object and enter an Expression using the Functions and Operators in the value field.

5.1.8.1. Arithmetic

abs(x) : absolute value

abs(-2) = 2

ceil(x) : rounded value to higher integer

ceil(1.2) = 2

ceil(1.8) = 2

floor(x) : rounded value to the lower integer

floor(1.2) = 1

floor(1.8) = 1

max(a,b) : max of a and b

max(0.1, 0.8) = 0.8

min(a,b) : min of a and b

min(0.1, 0.8) = 0.1

pow(a,b) : a at the power of b

pow(2,4) = 16

round(a) : rounded value to the closest integer

round(1.2) = 1

round(1.8) = 2

sign(a) : -1 if a<0, +1 if a>=0

sign(-12) = -1

sign(0) = +1

sign(12) = +1

sqrt(a) : square root of a

sqrt(64) = 8

operators : -a, a%b, a*b, a+b, a-b, a/b, a^b

note : a^b = pow(a, b)

5.1.8.2. Logic

When using logic and conditions in the parser, (not 0) is true and 0 is false.

!a : not a

1 when a = 0

0 when a!=0

a!=b

1 if a!=b (i.e a=2, b=4)

0 if a==b

a&&b : logical AND

1 && 1 = 1

1 && 0 = 0

a<b 1 if a < b

0 otherwise

a>b 1 if a>b

0 otherwise

a==b 1 if a==b

0 otherwise

a>=b 1 if a>=b

0 otherwise

a<=b 1 if a<=b

0 otherwise

a || b : logical OR

1 || 0 = 1

1 || 1 = 1

a?b:c This one is a conditional statement. It translates as:

if (a is true) then b else c

You can put anything in a, b and c.

Example:

Creating an Expression "e" in a MultiBall and setting its value to

Pad.x?x:y

will mean that :

if Pad.x != 0 (i.e Pad is pressed), e=x

if Pad.x == 0 (i.e Pad is not pressed), e=y

You could for example map the e Expression to a control change that will be mapped to the x axis when pad is pressed, and y axis when pad is released

5.1.8.3. Others

clamp(a, min, max) : Constrain a between min max

clamp(Fader.x, 0.2, 0.3) =

0.2 if Fader.x<=0.2

0.3 if Fader.x>=0.3

Fader.x otherwise

rand() : Returns a random value between 0 and 1at each frame (one frame is 16ms, the parser evaluates every Expression at each frame and send out MIDI or OSC if a mapped value has changed)

range(a, min, max) : Stretch a Variable that normally goes from 0 to 1, to range [min, max]

range(Fader.x, 10, 100) =

10 if Fader.x == 0

100 if Fader.x == 100

between 10 and 100 otherwise

5.1.8.4. Trigonometric

acos, asin, atan, cos, sin, tan, log, log10, exp are the same as their mathematical counterparts.

angle(x,y) : Returns the angle in radians formed by a vector of coordinates (x,y) to the positive x axis

Example:

Create a RingArea

Create a variable inside : a = angle(x-0.5, y-0.5)

this returns the angle position of the ball from the centre of the RingArea

norm(x, y) : Length of a vector of coordinates (x,y)

Example :

create a Multiball with 2 balls

variable inside : distance = normal(x[1]-x[0], y[1]-y[0])

this returns the distance between ball 0 and ball 1

5.1.8.5. Vector

fill(a, value, size) : Returns a vector with "size" items

the vector is filled from the left with n items = "value"

the number of items set to "value" depends on the a argument : a threshold between 0 and 1

when a = 0, there's no filling at all, and the vector is full of zeros

when a = 1, the vector is completely filled with items = value

when a = 0.5, the vector is half filled

Examples :

fill(1, 0.524, 4) = {0.524, 0.524, 0.524, 0.524}

fill(0, 0.524, 4) = {0, 0, 0, 0}

fill(0.5, 0.524, 4) = {0.524, 0.524, 0, 0}

firstof(a) : Returns the position of the first non-null item in a vector

this is often used with switches in radio mode : firstof(x) returns the position of the enabled switch in the matrix

Examples:

firstof({0,0,1}) = 2

firstof({1,0,0}) = 0

if the vector only contains null items, the function returns the size of the vector

firstof({0,0,0}) = 3

replace(a, b, position) : Takes an array, and replace the items starting at "position" with b

b can be an array or a singleton

Examples:

replace({0,0,0,0}, {12,2}, 0) = {12, 2, 0, 0}

replace({0,0,0,0}, 0.15, 3) = {0, 0, 0, 0.15}

If position is not an integer, the parser converts it using the floor function (next lower integer)

set(a, value, position) : Takes an array, and change the item at "position" to "value"

position can be an array or a singleton

Examples:

set({0,0,0,0}, 12, 0) = {12, 0, 0, 0}

set({0,0,0,0}, 12, {0,2}) = {12, 0, 12, 0}

If position is not an integer, the parser converts it using the floor function (next lower integer)

sizeof(a) : Returns the size of an vector (or 1 if applied on a singleton)

stretch(a, size) : Stretches a value or a range and returns a vector

Explanation :

If a is a single value (or a vector with a single item), the function returns a vector with "size" items containing that value

stretch(0, 12) = {0,0,0,0,0,0,0,0,0,0,0,0}

After that, you could do :

set(stretch(0, 12), 1, time%12)

This returns:

{1,0,0,0,0,0,0,0,0,0,0,0} when time<1

{0,1,0,0,0,0,0,0,0,0,0,0} when time<2

{0,0,1,0,0,0,0,0,0,0,0,0} when time<3

etc.

If a is a vector with more than one value, the function stretches the range over "size" items :

stretch({1,4}, 4} = {1,2,3,4}

stretch({0,10},11} = {0,1,2,3,4,5,6,7,8,9,10,11}

sumof(a) : Sums all the items of an array

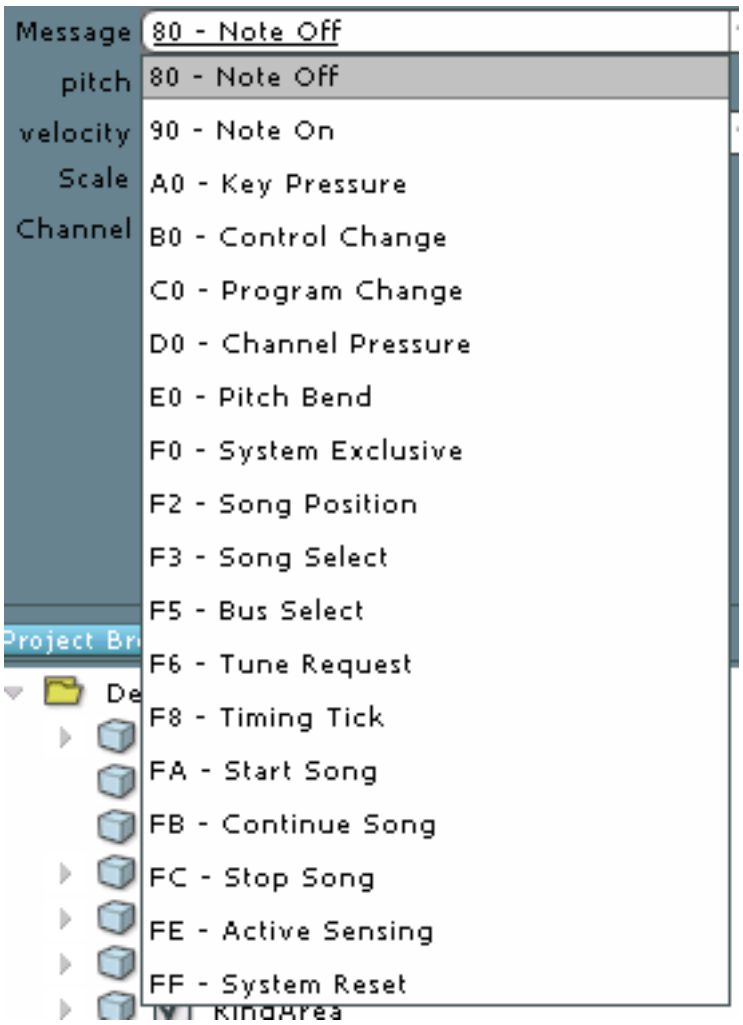
Chapter 6 MIDI Message Reference

Depending on the Message type you choose from the Message menu on the MIDI panel you get a different set of parameters.

6.1.1. The MIDI Messages

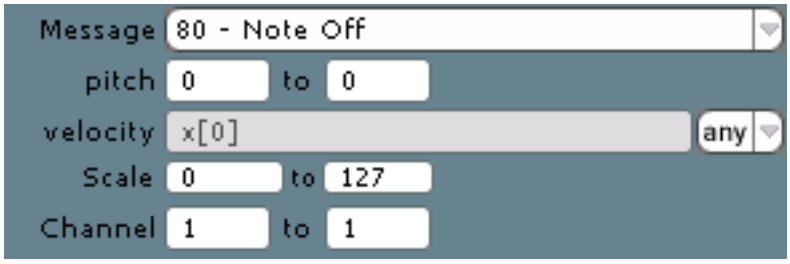
The descriptions given below are the standard definitions as given by the MIDI specifications. Of course it is up to you to use the various messages to your liking. In a world of flexible software the MIDI messages tend to lose their original meanings. In modular software like Reaktor or Max/MSP you can use any MIDI message to trigger or change anything you want.

We only give the standard MIDI panels as examples, here. Please note that with Custom MIDI messages the different parameters of the MIDI messages (like pitch, velocity and MIDI channel) can be controlled by any expression you want. Please read chapter 3.5.5 for details.



6.1.1.1. 80 – Note Off

The Note Off message is sent when a key is released.



pitch: Dial in the MIDI note number. If the value is a vector the second field will be extrapolated automatically. You can always override this setting by putting in your own value.

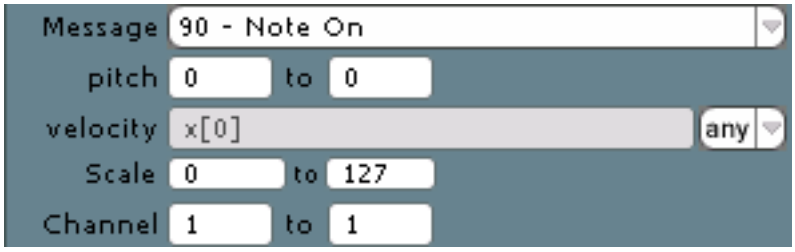
velocity: The Variable chosen from the Variable menu that is converted to velocity values is shown here.

Scale: Type in the desired target scaling for the MIDI messages. Possible values range from 0 to 127.

Channel: Dial in the MIDI channel. If the value is a vector the second field will be extrapolated automatically. You can always override this setting by putting in your own value. Possible values range from 0 to 16.

6.1.1.2. 90 – Note On

The Note On is sent when a key is depressed.



pitch: Dial in the MIDI note number. If the value is a vector the second field will be extrapolated automatically. You can always override this setting by putting in your own value.

velocity: The Variable chosen from the Variable menu that is converted to velocity values is shown here.

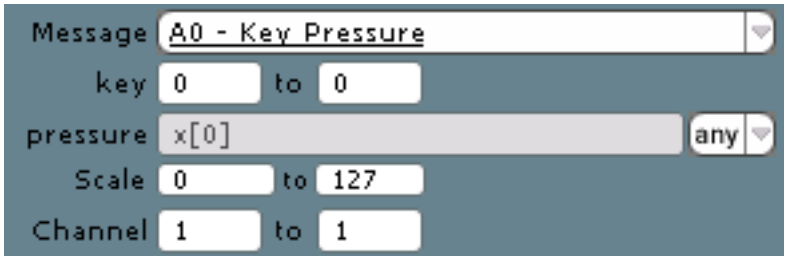
Scale: Type in the desired target scaling for the MIDI messages. Possible values range from 0 to 127.

Channel: Dial in the MIDI channel. If the value is a vector the second field will be extrapolated automatically. You can always override this setting by putting in your own value. Possible values range from 0 to 16.

6.1.1.3. A0 – Key Pressure (Polyphonic Aftertouch)

The Key Pressure, also called polyphonic aftertouch, gives the keyboarder a continuous controller for every key he had on his keyboard. It is used via pressing down on the different keys.

This works great for spreading out vector values to multiple key pressure values.



key: Dial in the MIDI note number. If the value is a vector the second field will be extrapolated automatically. You can always override this setting by putting in your own value.

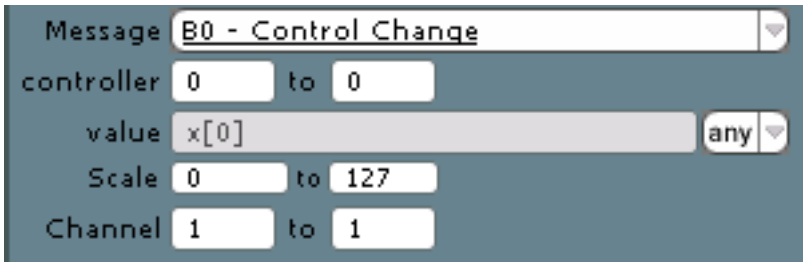
pressure: The Variable chosen from the Variable menu that is converted to key pressure values is shown here.

Scale: Type in the desired target scaling for the MIDI messages. Possible values range from 0 to 127.

Channel: Dial in the MIDI channel. If the value is a vector the second field will be extrapolated automatically. You can always override this setting by putting in your own value. Possible values range from 0 to 16.

6.1.1.4. B0 – Control Change

The Control Change data is used to produce pseudo-continuous data (ranging from 0 to 127).



controller: Dial in the MIDI controller number. If the value is a vector the second field will be extrapolated automatically. You can always override this setting by putting in your own value. Possible values range from 0 to 127

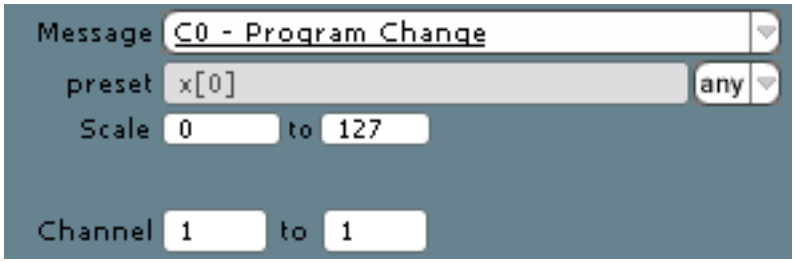
value: The Variable chosen from the Variable menu that is converted to the controller values is shown here.

Scale: Type in the desired target scaling for the MIDI messages. Possible values range from 0 to 127.

Channel: Dial in the MIDI channel. If the value is a vector the second field will be extrapolated automatically. Possible values range from 0 to 16.

6.1.1.5. C0 – Program Change

Used to change the patch number of synthesizers.



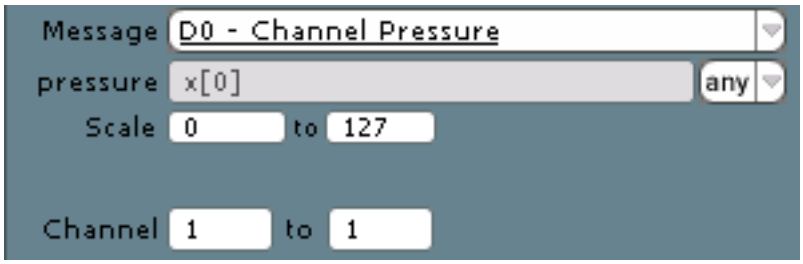
preset: The variable chosen from the Variable menu that is converted to program change values is shown here.

Scale: Type in the desired target scaling for the MIDI messages. Possible values range from 0 to 127.

Channel: Dial in the MIDI channel. If the value is a vector the second field will be extrapolated automatically. Possible values range from 0 to 16.

6.1.1.6. D0 – Channel Pressure

Another continuous controller used on a per MIDI channel basis.



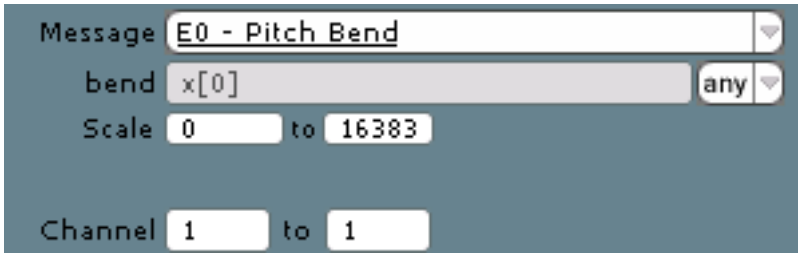
pressure: The Variable chosen from the Variable menu that is converted to channel pressure values is shown here.

Scale: Type in the desired target scaling for the MIDI messages. Possible values range from 0 to 127.

Channel: Dial in the MIDI channel. If the value is a vector the second field will be extrapolated automatically. Possible values range from 0 to 16.

6.1.1.7. E0 – Pitch Bend

Used to indicate a change in the pitch wheel. It is a 14 bit value providing a higher resolution than control changes.



bend: The Vvariable chosen from the Variable menu that is converted to program change values is shown here.

Scale: Type in the desired target scaling for the MIDI messages. Note that pitch bend values range from 0 to 16383.

Channel: Dial in the MIDI channel. If the value is a vector the second field will be extrapolated automatically. Possible values range from 0 to 16.

6.1.1.8. F0 – System Exclusive

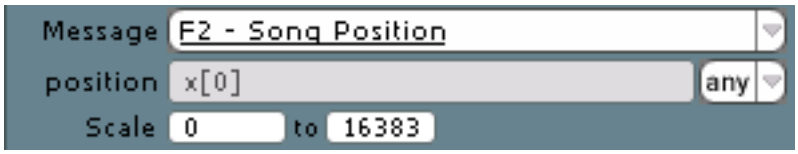
Used for custom data that is not covered by normal MIDI messages. The first part is usually a 7 bit value representing an ID. If the device or software recognizes the ID as its own, it will listen to the rest of the message. Otherwise, the message will be ignored.

trigger: The Variable chosen from the Variable menu that is converted to SysEx messages is shown here.

data: Type in the desired SysEx string you want to send. Please have a look at the MIDI Target's documentation for details about possible SysEx messages.

6.1.1.9. F2 – Song Position

A 14 bit value that holds the number of MIDI beats (1 beat= six MIDI Timing Tick) since the start of the song.

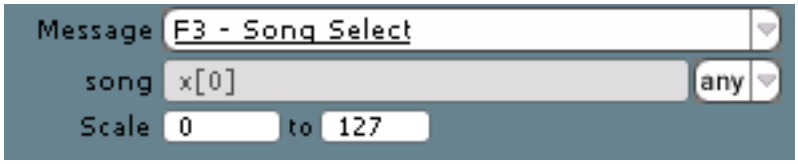


position: The Variable chosen from the Variable menu that is converted to song position values is shown here.

Scale: Type in the desired target scaling for the position pointer. Note that song position pointer values range from 0 to 16383.

6.1.1.10. F3 – Song Select

The Song Select specifies which sequence or song is to be played.

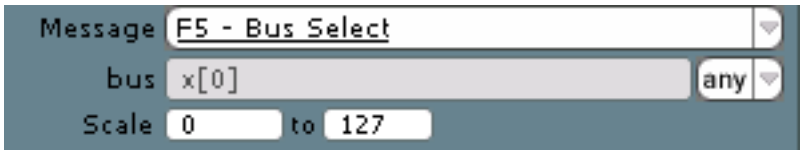


song: The Variable chosen from the Variable menu that is converted to song number values is shown here.

Scale: Type in the desired target scaling for the song number. Possible values range from 0 to 127.

6.1.1.11. F5 – Bus Select

The Bus Select message specifies to which MIDI output further data should be sent.



bus: The Variable chosen from the Variable menu that is converted to bus number values is shown here.

Scale: Type in the desired target scaling for the bus number. Possible values range from 0 to 127.

6.1.1.12. F6 – Tune Request

Used to trigger the tuning of oscillators in analog synthesizers.

trigger: The Variable chosen from the Variable menu that is used to trigger tune request messages is shown here.

6.1.1.13. F8 – Timing Tick

Sent 24 times per quarter note for synchronization purposes. Build your own MIDI clock.

trigger: The Variable chosen from the Variable menu that is used to trigger timing tick messages is shown here.

6.1.1.14. FA – Start Song

Used to start the song or sequence from the beginning.

trigger: The Variable chosen from the Variable menu that used to trigger start song messages is shown here.

6.1.1.15. FB – Continue Song

Used to start the song or sequence from where it was stopped.

trigger: The Variable chosen from the Variable menu that is used to trigger tune request messages is shown here.

6.1.1.16. FC – Stop Song

Used to stop the song or sequence.

trigger: The Variable chosen from the Variable menu that is used to trigger stop song messages is shown here.

6.1.1.17. FE – Active Sensing

When initially sent, the receiver will expect to receive another Active Sensing message each 300ms (max), or it will be assume that the connection has been terminated. At termination, the receiver will turn off all voices and return to normal (non-active sensing) operation.

trigger: The Variable chosen from the Variable menu that is used to trigger active sensing messages is shown here.

6.1.1.18. FF – System Reset

Resets all receivers in the MIDI chain to power-up status.

trigger: The Variable chosen from the Variable menu that is used to trigger system reset messages is shown here.

Chapter 7 APPENDICES

7.1. Network Configuration Guide

The Lemur control surface is a network device using the Ethernet protocol to connect and communicate with other devices on the network: Computers, routers, DHCP servers....

Thus you have to setup your Lemur and your network devices properly. Generally, a device can have multiple network adapters, like Ethernet, WIFI, Airport, USB Network Adapters, Firewire and others. Each adapter can have its own IP settings and it is highly recommended to set different sub-networks to avoid network problems.

7.1.1. Glossary

IP Address a unique sequence of 4 numbers witch identify each device on a network. An IP address can be setup in static mode or attributed dynamically by a DHCP server. An IP address is composed by 4 numbers between 0 and 255 as follows: 192.168.2.2 or 10.0.0.35.

Network Mask an IP-like sequence witch defines a sub-network of compatible IP addresses. Most of the time we use 255.255.255.0 or 255.255.0.0, which means that only the 3 or 2 first IP numbers should be equal to be compatible.

Gateway the IP address of the machine giving you access to IPs which are not included in the sub-network. In brief, a Gateway enables you to gain access to the internet from an intranet.

MAC Address a sequence of 6 numbers identifying each network adapter on a network. The MAC Address is unique and constant for each device.

7.1.2. Computer directly connected to a Lemur (as known as P2P)

Using a simple crossover RJ45 cable, you can directly connect your Lemur to any computer.

7.1.2.1. Computer configured as a DHCP Server – Connection Sharing

If your computer is configured as a DHCP Server, it will attribute dynamically an IP address to the Lemur. Most of the time your computer forces its IP address to 169.254.X.1 and the Lemur will obtain an IP like 169.254.X.X.

7.1.2.2. Computer and Lemur using DHCP

When the Lemur and the computer are set up as DHCP clients, they both wait for a DHCP server until the OS gives a default IP address like 169.254.X.X.

7.1.2.3. Computer and Lemur using Static IP

The most common way to configure a network without a DHCP server is to set up each device with different but compatible IPs using the same network mask. Most people use IP addresses like 192.168.0.X or 10.0.0.X.

Eventually, when using connection sharing, the computer can share its internet connection and give the Lemur access to another sub-network by using the Gateway Address.

7.1.3. Computers and Lemur connected to a Network

7.1.3.1. Network with a DHCP Server

Most network sharing devices allow multiple connections and manage the attribution of IP addresses through the DHCP protocol. These devices include WIFI/Ethernet Routers, Airport Base Stations and ADSL/Cable Modems with routers.

You only need to configure all your devices - including the Lemur - to use the DHCP mode.

7.1.3.2. Network without DHCP Server

If you have multiple devices connected through a hub or switch, you may have to configure them individually with a unique, static IP sharing the same network mask.

Use IPs like: 192.168.0.X or 10.0.0.X

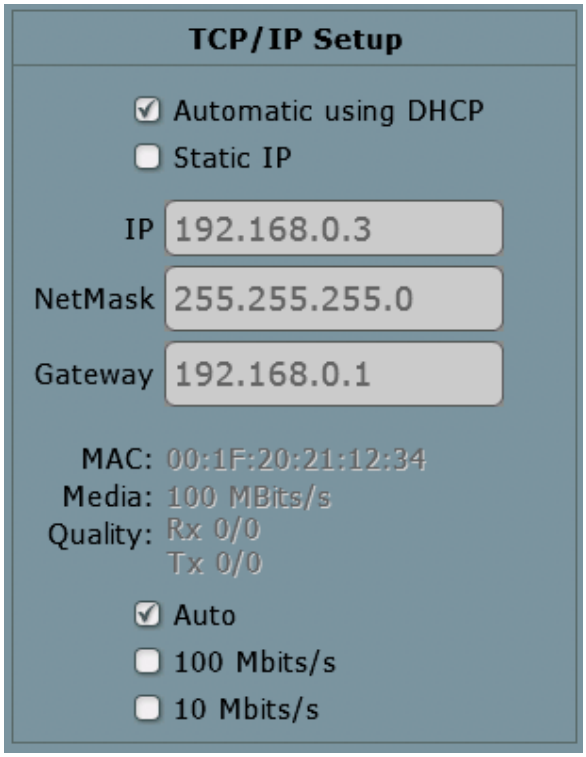
Mask: 255.255.0.0

7.1.4. Lemur Network Setup

You open the Network Settings via the Settings button, which is the left-most hardware button on top of the Lemur.

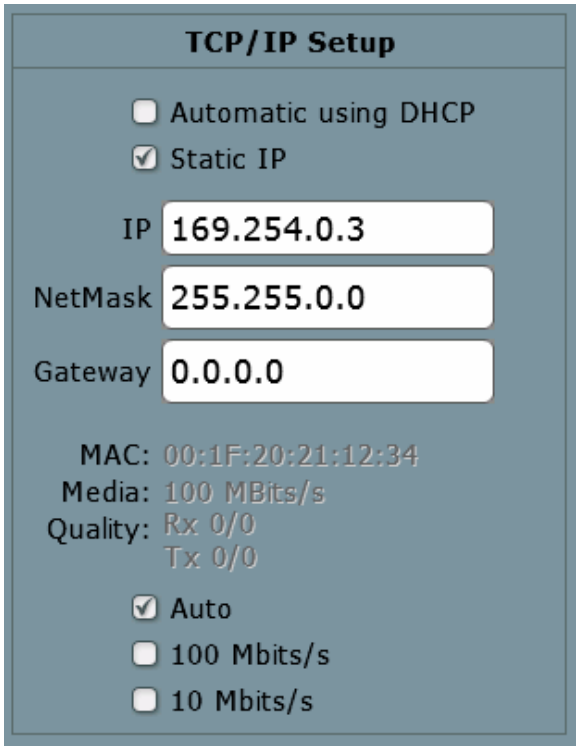
DHCP IP Configuration:

- Select "Automatic using DHCP".
- If a DHCP server is available, the Lemur will obtain a DHCP IP address automatically. If there is no DHCP server available, the Lemur will choose its own address.



Static IP Configuration:

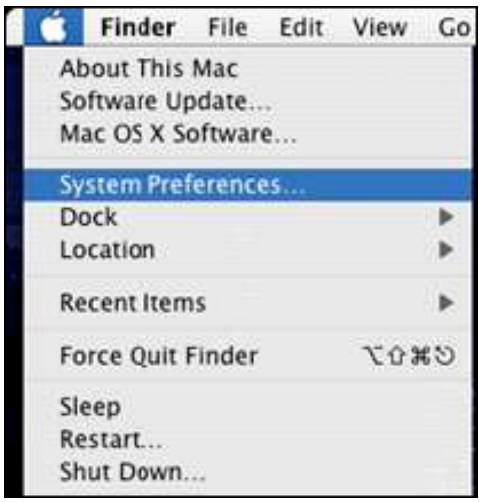
- Select “Static IP”.
- Enter assigned static IP address into the IP field.
- Enter NetMask (255.255.0.0)



7.1.5. MacOS Network Setup

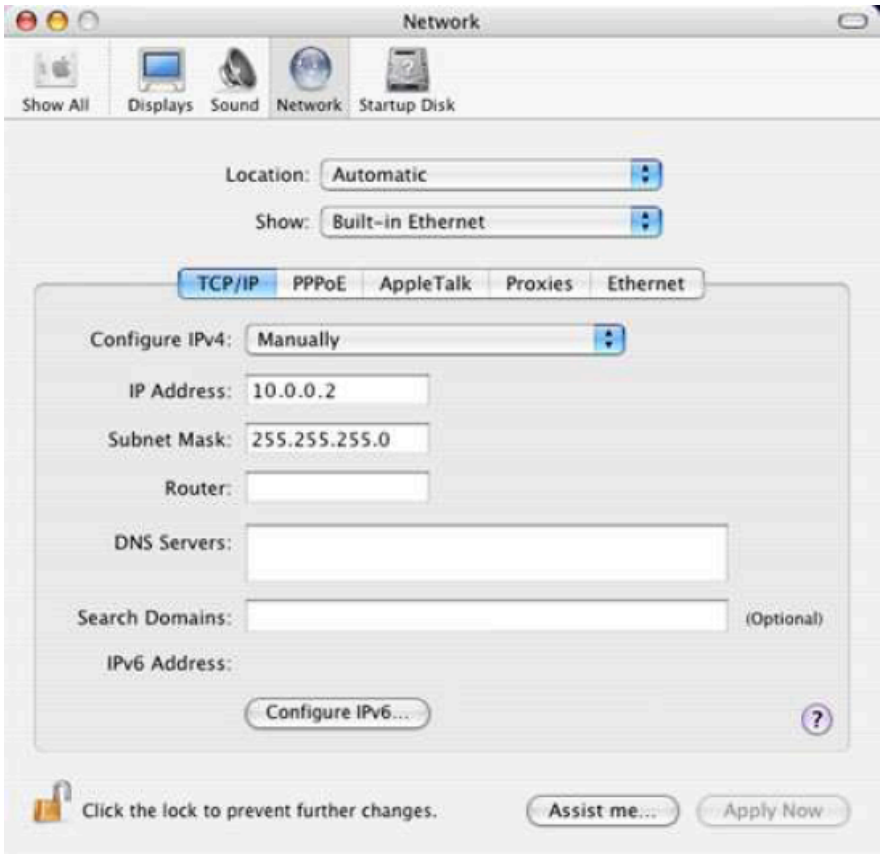
Static IP Configuration using Mac OS X:

- Go to the Apple menu and select System Preferences.



- Under Internet and Network, select Network.

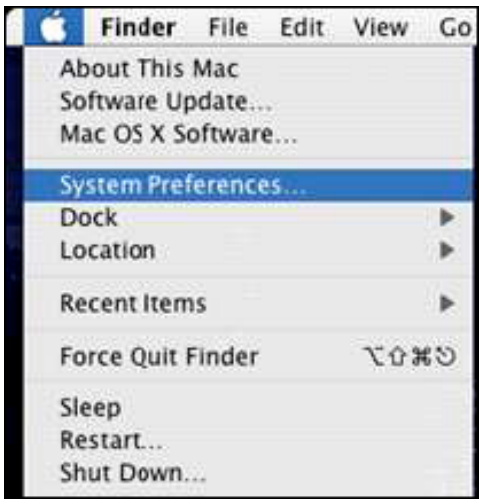
System Preferences
Show All
Displays
Sound
Network
Startup Disk
Personal
Appearance
Desktop & Screen Saver
Dock
Exposé
International
Security
Hardware
CDs & DVDs
Displays
Energy Saver
Keyboard & Mouse
Print & Fax
Sound
Internet & Network
.Mac
Network
QuickTime
Sharing
System
Accounts
Classic
Date & Time
Software Update
Speech
Startup Disk
Universal Access


Network
Show All
Displays
Sound
Network
Startup Disk
Location: Automatic
Show: Built-in Ethernet
TCP/IP
PPPoE
AppleTalk
Proxies
Ethernet
Configure IPv4: Manually
IP Address: 10.0.0.2
Subnet Mask: 255.255.255.0
Router:
DNS Servers:
Search Domains:
(Optional)
IPv6 Address:
Configure IPv6...
Click the lock to prevent further changes.
Assist me...
Apply Now

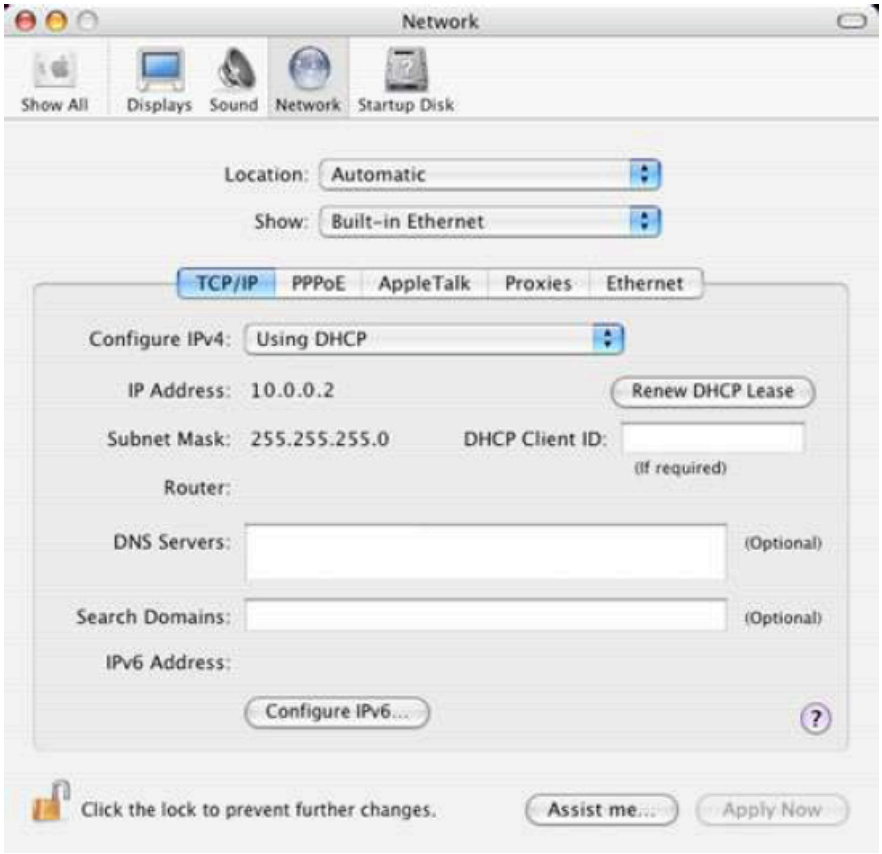
- Change the **Show** pop-up option to **Built-in Ethernet**.
- Under **Location**, choose **Automatic** (or name the new connection).
- Under **Show**, Select **Built-in Ethernet**.
- Change to the **TCP/IP** tab.
- Set **Configure IPv4** to **Manually**.
- Enter assigned static **IP Address**.
- Enter **Subnet Mask** 255.255.255.0
- Click the **Apply** Now button, and **quit** the System Preferences application.

DHCP Configuration using Mac OS X:

- Go to the Apple menu and select System Preferences.



- Under Internet and Network, select Network.



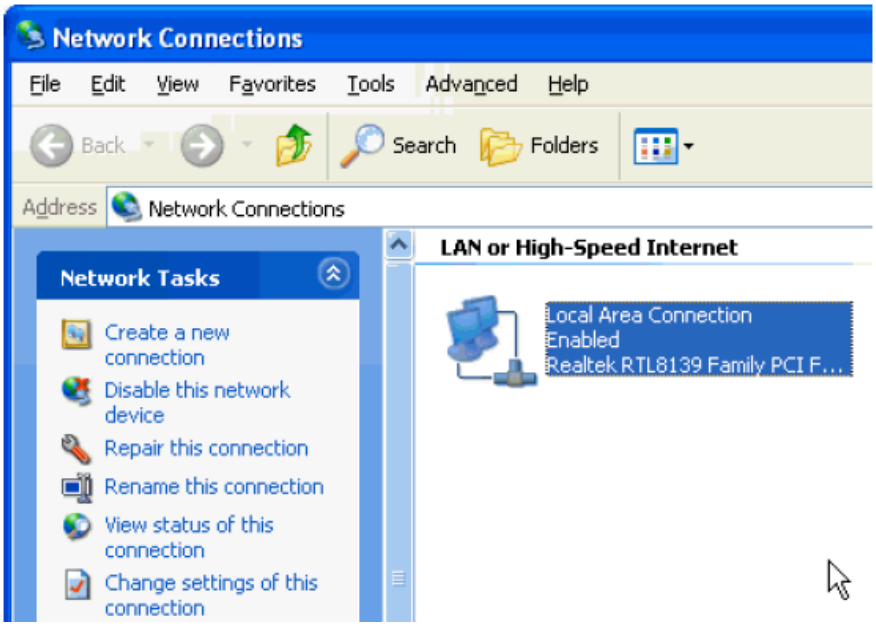
- Change the **Show** pop-up option to **Built-in Ethernet.**

- Under **Location**, choose **Automatic** (or give this connection a name).
- Under **Show**, Select **Built-in Ethernet.**
- Change to the **TCP/IP** tab.
- Set **Configure IPv4** to **Using DHCP**.
- Click the **Apply Now** button, and **quit** the System Preferences application.

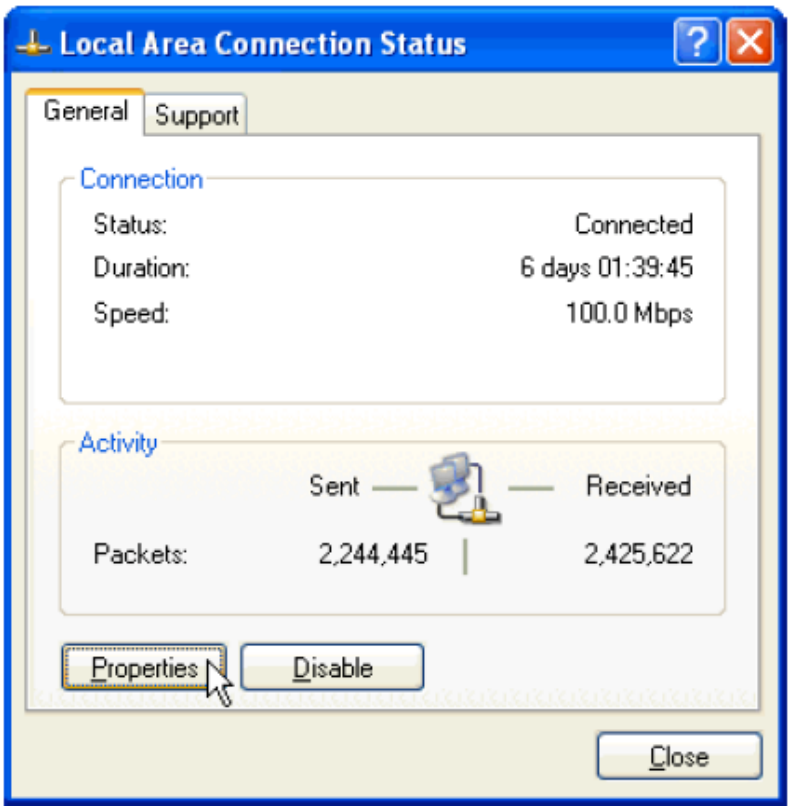
7.1.6. Windows™ XP Network Setup

- On the Desktop, click right on the **My Network Places** icon and select **Properties**.

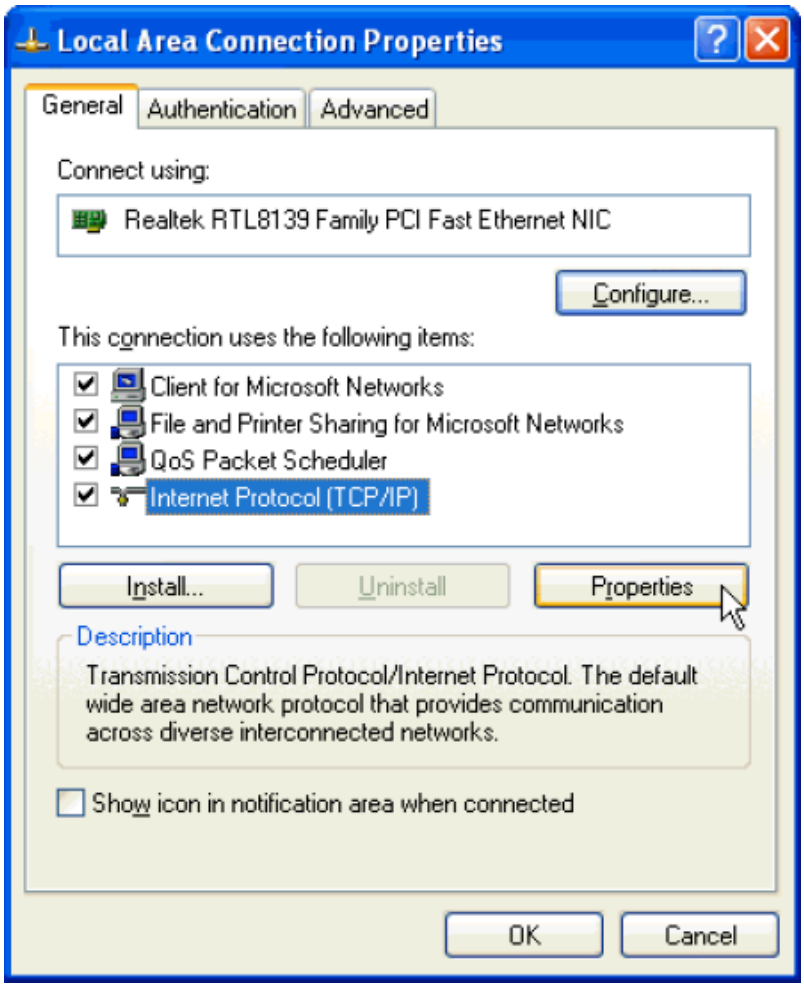
- Select your **LAN connection** and double-click on it.

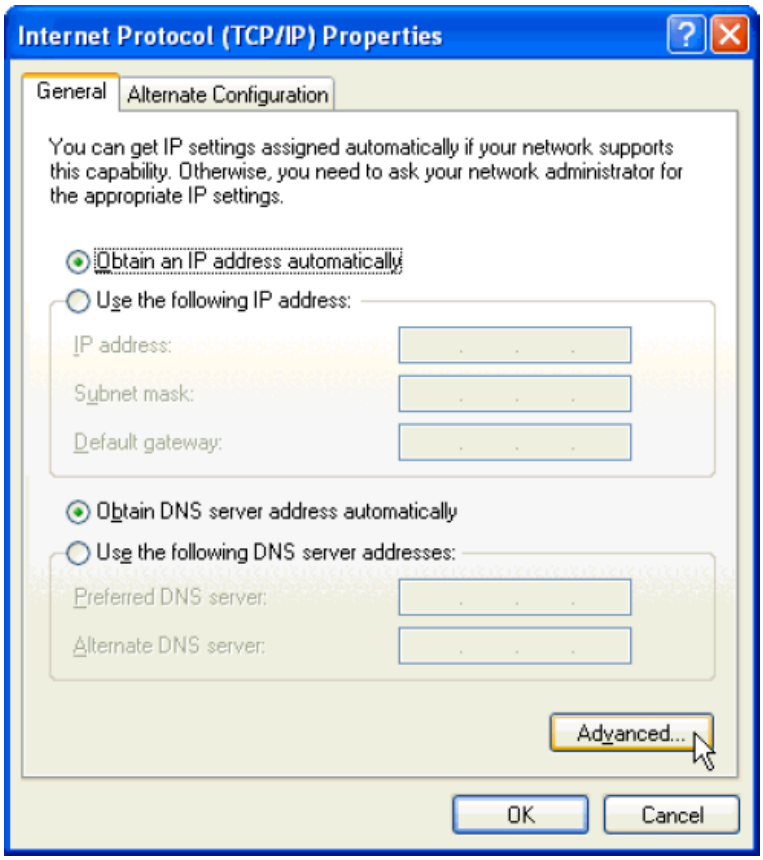


- Click on the **Properties** button.



- Select **Internet Protocol (TCP/IP)** and click the **Properties** button


Local Area Connection Properties
General
Authentication
Advanced
Connect using:
Realtek RTL8139 Family PCI Fast Ethernet NIC
Configure...
This connection uses the following items:
Client for Microsoft Networks
File and Printer Sharing for Microsoft Networks
QoS Packet Scheduler
Internet Protocol (TCP/IP)
Install...
Uninstall
Properties
Description
Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.
Show icon in notification area when connected
OK
Cancel

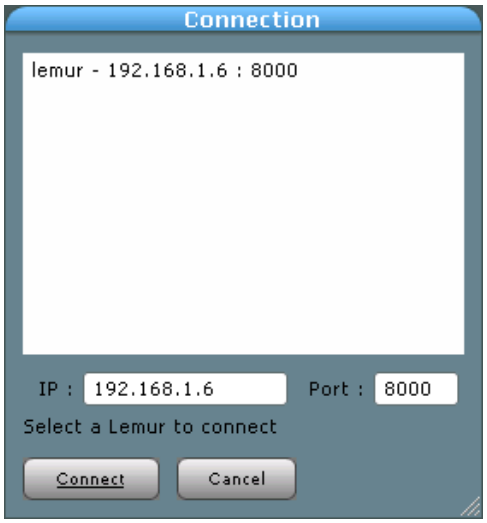

Internet Protocol (TCP/IP) Properties
General
Alternate Configuration
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.
Obtain an IP address automatically
Use the following IP address:
IP address:
Subnet mask:
Default gateway:
Obtain DNS server address automatically
Use the following DNS server addresses:
Preferred DNS server:
Alternate DNS server:
Advanced...
OK
Cancel

- Select **Obtain an IP address automatically** to use DHCP configuration or select **Use the following IP address** to use static IP configuration. Enter assigned static IP address and Subnet Mask.
- Click **OK** to confirm changes.

7.1.7. Lemur – Computer network communication

To confirm that your network setup is correct, power on the Lemur and open the **JazzEditor**.

Click the **Connect** button. You must see the Lemur on the list.

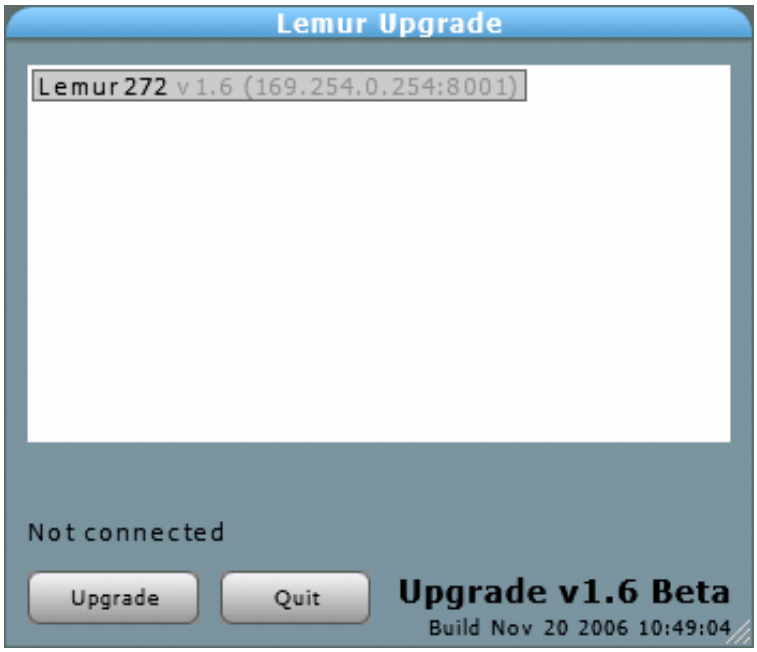


7.2. Shortcuts

Command	Keyboard shortcut
New Project	Control+N
Save Project	Control+S
Import Library	Control+I
Copy	Control+C
Paste	Control+V
Redo	Control+Y
Lemur Synchro	Control+R
Lemur Settings	Alt+S
Create Object	Control+Shift+O
Create Custom MIDI	Control+Shift+M
Open Project	Control+O
Save Project As	Control+ Shift+S
Export Module	Control+E
Cut	Control+X
Undo	Control+Z
MIDI Map	Alt+M
Lemur Connection	Alt+C
Create Interface	Control+Shift+I
Create Expression	Control+Shift+E
Create Custom MIDI	Control+Shift+M
Toggle to Run Mode	Hold E

7.3. Upgrade your Lemur

The Lemur is fully **reprogrammable** by upgrading its **firmware**. You will always find the most recent upgrade on our **website**. An upgrade can contain bug corrections or new features.

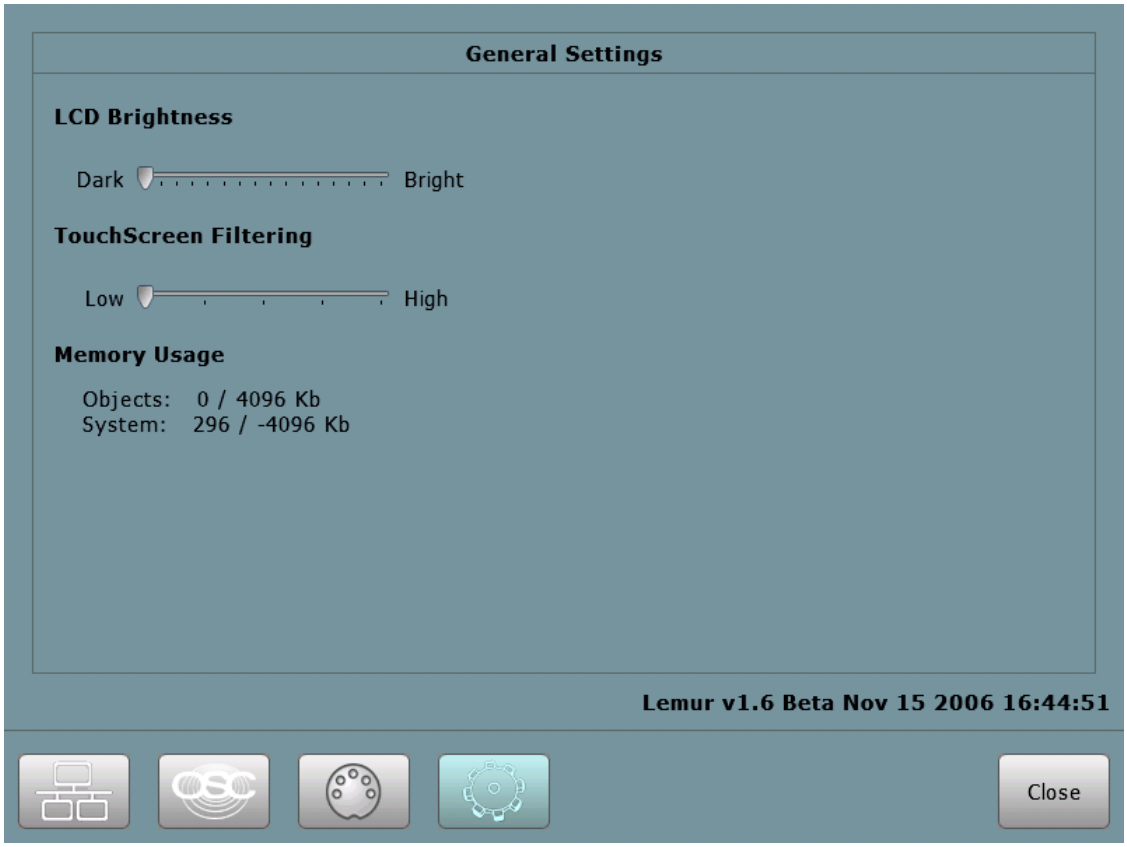


To upgrade your Lemur to a a different firmware than the one it's running, download the corrsponding **Lemur Updater** application, and run it as you would run the JazzEditor, with your Lemur on and connected. Select the Lemur you want to upgrade in the List and click **Upgrade.** The new firmware will be uploaded to the Lemur. The Lemur will then reprogram itself and reboot with the new firmware.

Should the upgrade process fail, you can boot the Lemur in **factory mode** and try again. To boot in **factory mode**, turn off the Lemur and switch on while holding the leftmost button.

If all things fail, contact your reseller for help.

7.3.1. Lemur hardware settings



The leftmost button on your Lemur opens the **Hardware Settings**. Choose between **Network Settings** (discussed in Chapter 2 and in the Appendix about Network Configuration) , **OSC Settings** (discussed in chapter 3.3), **MIDI Settings** (also to be found in chapter 3.3) and the **Screen Settings**.

The Screen Settings are opened by touching the **Settings Button** on the Hardware Settings Window.

There are two sliders for controlling you Lemur's screen hardware:

- **LCD Brightness** controls the brightness of your Lemur's screen. It may be useful to dim the display if you want to work under dark stage conditions where too much light may not be desirable.
- The **TouchScreen Filtering** is for controlling the sensitivity of the Lemur's touch screen. If you experience multiple triggering or other touch artefacts, try a higher Filtering level.

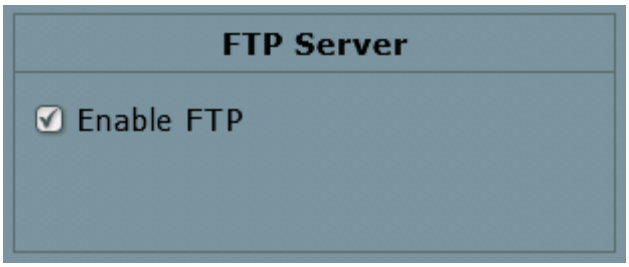
7.4. The FTP Server

Yep, the Lemur runs an **FTP server**. This can be helpful for uploading large numbers of Projects to the Lemur without having to channel them through the JazzEditor.

It can also come in very handy for retrieving Projects that you lost due to a mishap with the JazzEditor, e.g. if you accidentally closed it without saving. Simply save the Project to the internal memory of the Lemur, download it via FTP to your computer and everything is cool

again. As we don't want everybody on the Net work to log into our Lemur and fill it up with trash, the FTP server is deactivated by default.

Push the Settings button on your Lemur and you are presented with various Setup options. Touch the **Enable FTP** flag to activate the FTP server.



Now you can use you favourite **FTP client** to connect to the Lemur on its IP address. A web browser also does the job.

Index von ftp://169.254.1.99/ - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
ftp://169.254.1.99/
Index von ftp://169.254.1.99/
In den übergeordneten Ordner wechseln
AllObjects.jzml 29 KB 01.01.2006 00:00:00
Tee2.j
Notron
Keyboa
C:\WINDOWS\system32\ftp.exe
Verbindung mit 169.254.1.99 wurde hergestellt.
220 Welcome on the Lemur Server
Benutzer (169.254.1.99:(none)): lemur
331 User Login OK, waiting for password
Kennwort:
230 User Password OK, Server ready
ftp> dir
200 Set to Active mode
150 Active Data connection
-rwxrwxrwx 1 user group 29671 Jan 1 2006 AllObject
-rwxrwxrwx 1 user group 74008 Jan 1 2006 Tee2.jzml
-rwxrwxrwx 1 user group 70615 Jan 1 2006 Notronish
-rwxrwxrwx 1 user group 14664 Jan 1 2006 Keyboard.
226 Directory listing successfully sent
FTP: 64d Bytes empfangen in 0,11Sekunden 2,51KB/s
ftp> put F:\Lemur\Salzgrube\salzig4.jzml
200 Set to Active mode
150 Active Data connection
226 File successfully received
FTP: 64d Bytes gesendet in 0,00Sekunden 61593000,00KB/s
ftp> dir
200 Set to Active mode
150 Active Data connection
-rwxrwxrwx 1 user group 29671 Jan 1 2006 AllObject
-rwxrwxrwx 1 user group 74008 Jan 1 2006 Tee2.jzml
-rwxrwxrwx 1 user group 70615 Jan 1 2006 Notronish
-rwxrwxrwx 1 user group 14664 Jan 1 2006 Keyboard.
-rwxrwxrwx 1 user group 61593 Jan 1 2006 salzig4.j
226 Directory listing successfully sent
FTP: 64d Bytes empfangen in 0,28Sekunden 1,22KB/s
ftp>

The **login** and **password** is arbitrary, use anything you like, the Lemur doesn't care. You have to type something, though. You can see the list of Projects saved on your Lemur. Download and upload them from / to your computer to your hearts content.

7.5. Warranty

7.5.1. Warranty and Liability

Please check the delivered products for obvious defects. By opening the original packaging or downloading the product you accept all copyrights as well as the warranty restrictions. Should you find any defect of the merchandise during the one-year warranty period, JAZZMUTANT will - at its sole discretion - either replace the merchandise or remove any defects. Should we be unable to do so within an acceptable period of time, you are entitled to cancel your order. We are liable for the culpable breach of fundamental contractual

obligations (so-called cardinal obligations) and the absence of any promised qualities. Otherwise, we are liable only within the limits set by applicable liability laws or - for whatever legal reason – when any damages were caused with intent or gross negligence by us or one of our affiliates. Neither our affiliates nor we are liable in any case not described above. In case of culpable breach of cardinal obligations without intent or negligence on our part, our affiliates or we will be liable only to the extent of typically foreseeable damages and not for damage not incurred to the delivered merchandise itself. The risk of loss is transferred to you once the merchandise has been sent to you. This

also applies with part deliveries. If you are buying software via electronic delivery (Instant Delivery), we carry the risk of loss until the software has been fully transferred to your computer.

7.5.2. Your Rights

The customer's rights concerning refunds of all or part of a payment are valid only when such rights were legally confirmed or were accepted by us in writing. The orderer's right to retention is valid only provided claims are based on the same contract. The preferred method of written communication is email and is held to be equal in status to letters transmitted through the postal service. The term "correspondence" below is taken to include email. Deadlines are taken, insofar as is not otherwise mutually agreed, to always mean the date of receipt by the recipient. As far as email is concerned, the date of receipt is considered to be the date of dispatch. If there is disagreement about the date of dispatch, the time of receipt is taken as the receipt of the message on the target email server.

7.5.3. Applicable laws and court of jurisdiction

The law of France applies in exclusion of UN buying law, even when the order is placed in another country or the order is to be delivered into another country. If you are resident outside France, or if your usual place of abode is outside France, the court of jurisdiction for all claims in relation to your order is in Bordeaux, France. In the event of a legal dispute, the official address is JAZZMUTANT, 45 cours Alsace Lorraine 33000 BORDEAUX, France.

Appendices

Index

-,64
,21,35,46,79
,21,35
+,64
Accessing variables,104
Active Sensing,118
ADSR+H,42,56
,28,39
,28,39
,28,39
,28,39
Alignment,81
any,64
Arithmetic,106
Attraction,40,56
Behavior,28,39,56,66,71
Behavior panel,28,71
Behavior Properties
  Fader,83
  Leds,85,86
  Monitor,87,100
  Multiball,89
  MultiSlider,91,95
  Pads,81,93
  SignalScope,97
  Switches,100
bi-directional,48,58
Bi-directional Control,75
Brightness ADSR,43,56
Bring to Front,38
Bus Select,116
,21,42,56
,23,63
Channel Pressure,115
Checkbox,68
Checkboxes,24
Color,38
Color Off,81
Color On,81
,23,31
Connection,23,30
Connection Sharing,119
Constant controller values,104
,21,44
  Creation,44
  Lock,45
,21
Continue Song,117
Control Change,114
Controller,62
Controller Mappings,61
Controlling sub-objects,66
Copy,23,27,37
,38
Create
  Custom MIDI mappings,26
  Expressions,26
  Objects,26
Create
  Interfaces,26
Create commands,26
Create Expression,72,76
,46
Crossover Ethernet cable,9,19
,21
,21
,23,57,59
Custom Button,80

Custom MIDI Messages,73
Cut,23,27
Daemon,13,20
Delete,27,65
DHCP IP Configuration,120
DHCP Server,119
Dimensions
  Monitor,86,100
  Multiball,88
  MultiSlider,91,94
  Pads,80,81,92
  RingArea,96
  SignalScope,97
Disclosure triangles,24,45,56
,23,31
down,64
Edit mode,27
Editing Area,27
Enable FTP,132
,21,43,56
Ethernet cable,9
Ethernet jack,9
Export,27
Export Selection,23
Expression,68,72
Expressions,12,21,103
Fader,38,82
firmware,130
Font,81
Font size,67
Frame,104
Friction,40,56,66
FTP client,132
FTP Server,131
Functions,12,21,76,103
  Arguments,76
  Arithmetic,106
  Logic,107
  Name,76
  Script,76
  Trigonometric,109
  Vector,109
Functions Reference,106
Gateway,119
General Properties
  Fader,82
  Leds,84,86
  Monitor object,86,100
  Multiball,88
  MultiSlider,91,94
  Pads,80,81,92
  RingArea,96
  RingArea,96
  SignalScope object,97
  Switches,98,99
Global,69,76
Grid,43,82
Group,27,38
Grouping,38
Hardware
  Project browser,8
,13,131
Hierarchy,45,76
Hold,42,56
Import,27
Import Library,23
Interacting Objects,66
,21,38
  Creation,32
  Name,33,56
  Order,33,56
Interface building,30

Interface List button,8,13
Interfaces,21
Interface List,10,21
Navigation buttons,10,21
Switch via OSC,10,21
Internal clock,104
Interpolate,40,56
IP Address,119
,56,57
,21
Toolbar,22
Workspace,21
,21
JazzEditor.dmg,15
JazzEditor_setup.exe,16
,21,35
,21,35
Key Pressure,113
Keyboard Shortcuts,129
Label,36
Layout,39
,13,131
Leds,85
Lemur
MIDI Settings,50
Project Browser,65
Syving Projects,65
,21
Lemur Network Setup,120
Lemur Project browser,8,10
,21
Lemur Synchronization,23
LFO,106
Load at Startup,65
,23,32
Local Variable,68,71
Lock,45
Logic,107
,21,56
DHCP,123
Network Setup,121
Static IP,121
,21,55
MAC Address,119
Mac OS Software Installation,15
Maintenance,14
,28,39
,28,39
Mass-Spring,40,56
Max/MSP,48
Memory Display,23
,29,59
,23,63
Bi-directional,75
,23,50
,23,50
,23,50
MIDI assignment,62
,23,63,75
,21
MIDI Daemon,20
Mac,21
MIDI port browser,51
Setup,51
MIDI data,20
,21
MIDI Map window,61
MIDI Mapping,23,60
,23,63

,23,63
,23,63
,23,63
,23,63
,23,63
,23
,23,63
MIDI Message Reference,112
MIDI Messages,112
MIDI Note Messages,62
,21
MIDI Panel,29,60
Automatic mapping,62
Controller,62
Message,61
MIDI ports,20
,21,49,56
MIDI Targets,13,21,23,29,49,50,59
,21,56
Modules,12,21,46,79
Import/Export,46,79
Monitor,86,100
,23,63
MultiBall,42,56,88
Speed,41,56
Multiball ADSR,90
MultiSlider,41,56,91,94
Name,36
Name conflict,37,38
Navigation,8,13
Network configuration,18,119
Network configuration - direct connection,18
Network Mask,119
New Project,22
Non-crossover Ethernet cable,9
Note Off,113
Note On,62,113
,21
Creation,33
Object Reference,79
Objects,11,21
Custom Button,80
Fader,82
Hierarchy,46,76,79
Interaction,66
Layout,39
Leds,85
Monitor,86,100
Multiball,88
MultiSlider,91,94
Overlap,27
Pads,80,92
Resize,27
RingArea,95
SignalScope,97
Switches,99
Transparency,27
Open,65
Open Project,23
Open Sound Control,46
Operators,12,21,102
Operators Reference,106
Bi-directional,75
,23
,23,48
Setup,46
,23
Variables menu,57
OSC address,37,38
OSC Data
Pads,80

OSC mappings,57
OSC Panel,29,57
OSC Targets,13,21,23,47
Pads,62,92
Pads ADSR,93
,23,63
Parser Reference,102
Paste,23,27,37
Physics,39,56
  Attraction,40,56
  Friction,40,56
  Tension,42,56
Physics menu,40,56
  Interpolate,40,56
  Mass-Spring,40,56
  None,40,56
Pitch,74
Pitch Bend,115
,23,31
Polyphonic Aftertouch,113
,23,48
ports,49

  ,56,57
Power Cable,14
Power Input,9
Power Supply,9
Power Switch,9
Precision,69
Program Change,114
,21
  hierarchy,45
  ,38
Project Browser
  Checkboxes,24
  Functions,26
  Global Variables,26
  Objects,24
  Operators,26
  Variables,24
Project browser,8,24,65
  Colors,26
  Interfaces,24
,21
Projects
  Saving,35
  Transfer,35
Properties,27,28,37,38
  modifying,40,56
  Precision,69
  value,69
Properties panel,27
Range Function,75
Reaktor 5,49
Rear Panel,9
Redo,23
Resize,27,36
RingArea,95
Run mode,27
Save As,23
Save Current,65
Save Project,23
Saving,35
Scale,75
Script,29
Script panel,29
Send mode,56
Send to Back,38
Settings,23
Settings button,8,13
Shortcuts,129
SignalScope,97

Singletons,102
Slider,41,56
Software & Hardware Requirements Mac,15,16
Software Installation,15
Song Position,116
Song Select,116
Speed,41,56
Start Song,117
Static IP,119
Static IP Configuration,120
,23,31
Stop Song,118
Style Off,81
Style On,81
SurfaceLCD,98
Switches object,99
Switching Interfaces,76
Synchronization,23,34
System Exclusive,116
System Reset,118
,23,63
Targets,13,21
Tension,42,56
Time,105
Time Code,105
Timing Tick,117
,21,22
,13,131
Transparency,27
Transparent,68
Trigger Mode,74
,29,64,75
Trigonometric,109
Tune Request,117
Undo,23
Ungroup,27,39
up,64
up and down,64
Upgrade,130
,23
,23
Value Coloring,102
Value flag,69
,21,56,68
Variables,21
,29,58,60
Global,12,21
Hierarchy,76
Leds Object,84,86
Local,12,21
Monitor,86,100
Multiball,88
Multi-dimensional,12,21
MultiSlider,91,94
Pads,80,92
RingArea,95
SignalScope,97
Switches,98,99
,29,57,59
Vector,71,109
Vector Variables,71
Vectors,78,102
Velocity,75
,21
,21,55
Warranty,133
,21,56
Network Setup,125
Windows XP Software Installation,16
WLAN,19
Workspace,21

Appendices

Workspace Configuration,29

x Variable,57